UNIVERSITA' DEGLI STUDI DI SIENA

FACOLTA' DI INGENGERIA

Corso di Laurea in Ingeneria Informatica

STUDIO ED IMPLEMENTAZIONE DI UN'ARCHITETTURA DISTRIBUITA PER LA DISTRIBUZIONE DI CONTENUTI MULTIMEDIALI

Relatore

Prof. Marco Maggini

Correlatore:

Dott: Cesare Bertoni

Tesi di Laurea di:

D'Aprile Roberto

Anno Accademico 2006/2007

Indice

Indice delle Immagini

Introduzione

L'obiettivo di questa tesi è lo sviluppo di applicazioni che sfruttino le architetture disponibili nel Laboratorio del Consorzio Operativo del Monte dei Paschi per migliorare l’efficienza nel lavoro di conversione di video digitali e nella distribuzione di contenuti multimediali. In particolare il lavoro svolto si suddivide in due parti, nella prima è stato studiato il problema dell’elaborazione di un video digitale, o più nello specifico della sua conversione dal formato DVD-Video, utilizzato per la sua memorizzazione su un supporto DVD, che sfrutta lo standard di compressione MPEG-2 ad un singolo file .avi utilizzando un codec che implementa la compressione MPEG-4 più semplice da memorizzare e riprodurre su un computer, nonché necessario per un ulteriore trasformazione del file in formato WMV (codec proprietario della Microsoft) correntemente utilizzato come standard nel Consorzio. Questo genere di elaborazione è estremamente pesante sia in termini di Hardware necessario e di potenza computazionale che in termini di tempo; ad esempio la codifica di un filmato della durata di circa un ora e mezza ottenuto da una videocamera (qualità media) può richiedere dalle 3 alle 6 ore di lavoro ed un uso intensivo del processore e della scheda grafica per tutto il tempo. Un modo per cercare di ovviare a queste richieste è la parallelizzazione del calcolo su più macchine, realizzata sfruttando l’architettura GRID presente nel laboratorio, che permette, una volta suddiviso il lavoro in una serie di processi, di distribuirli in modo batch sui nodi (terminali) ad essa collegati; questa soluzione non solo permette di diminuire il tempo richiesto ma ci garantisce anche una migliore gestione delle capacità computazionali della rete, assegnando il carico ai nodi che al momento hanno un minor uso di risorse. Il secondo problema affrontato riguarda la creazione di un sistema che permette una migliore gestione della richiesta di banda nella rete interna del Consorzio a fronte di numerose richieste di Streaming video; il problema consiste nel fatto che le richieste di streaming vengono tutte indirizzate al nodo centrale di una rete a stella, e nel caso si verifichino più richieste identiche il flusso di streaming viene duplicato, rischiando la saturazione della banda. Per tentare di risolverlo è stato creato un servizio che funge da punto di accesso intermedio e che si occupa di gestire le richieste in modo da evitare duplicazioni inutili degli stream; inoltre è stata sviluppata un’applicazione che si occupa di inviare le richie-

ste e permettendo di collegarsi a streaming già attivi e supportando una modalità di ritrasmissione dei flussi video in modo da poter creare un servizio di steaming peer to peer.

Nel primo capitolo introdurremo brevemente i concetti principali della rappresentazione di un video, partendo dai metodi di creazione e di visualizzazione, illustrandone caratteristiche e proprietà, mostrando alcuni esempi di standard per la trasmissione video, concludendo infine con alcuni cenni sulla rappresentazione audio. Ciò fornirà le basi per comprendere meglio i capitoli successivi.

Nel secondo capitolo ci focalizzeremo sul problema della compressione del video e dell’audio digitale, esplorando i concetti e gli strumenti alla base della riduzione di informazione, come la ridondanza spaziale o temporale e la compensazione del moto, per poi focalizzarci particolarmente sugli standard che sono stati utilizzati nel lavoro, cieè l’MPEG-2 e l’MPEG-4, illustrando brevemente gli algoritmi che sono alla base e le caratteristiche di questi standard.

Nel terzo capitolo esploreremo il procedimento alla base della conversione da uno standard di codifica all’altro, ponendo particolare attenzione alla descrizione dei vari strumenti software utilizzati nei vari passi del lavoro quali l’indicizzazione e l’editing auido e video.

Nel quarto capitolo parleremo dell’architettura GRID grazie alla quale è stato sviluppato il progetto, iniziando da un’analisi generale per poi descrivere la specifica tipologia applicata ed utilizzata nel laboratorio.

Infine nel quinto capitolo descriveremo le applicazioni create, gli esperimenti fatti, i problemi incontrati ed i risultati ottenuti.

Capitolo 1

Rappresentazione Video

Ormai da diversi anni la rappresentazione di contenuti multimediali in forma digitale ha ottenuto un posto in primo piano sia in ambito professionale che nella vita quotidiana, affermandosi grazie alle migliorie che comporta nell'ambito della conservazione e della duplicazione oltre che per le numerose nuove strade che apre nel campo della multimedialità. Alcuni esempi di ciò si vedono nell'ambito delle trasmissioni televisive e satellitari che stanno abbandonando la tecnologia analogica, anche in vista dell'imminente arrivo dell'HDTV (TV ad alta definizione), oppure nel campo della telefonia con l'introduzione della videofonia, od ancora in ambito domestico in cui si nota come gli strumenti di produzione, quelli di conservazione ed infine quelli di riproduzione siano quasi completamente migrati verso il digitale, abbandonando i vecchi strumenti come i nastri magnetici in favore dei nuovi supporti come CD, DVD, dischi rigidi e memorie flash, questi ultimi utilizzati per strumenti come i lettori MP3 o gli iPod che riuniscono le funzionalità di supporto e riproduzione, aumentando così la flessibilità dello strumento. Inoltre molte novità in campo multimediale sono state introdotte grazie allo sfruttamento della rete Web che ha permesso la creazione di sistemi di distribuzione multimediale estremamente competitivi come ad esempio le reti Peer to Peer o i servizi di streaming per la distribuzione dati da cui sono nati servizi come le chat video, gli streaming radiofonici o le più recenti trasmissioni di canali televisivi su Internet. I motivi principali del passaggio dall'analogico al digitale sono dovuti a vari fattori:

- Maggiore facilità di trasmissione e duplicazione: trasmettere un segnale analogico genera inevitabilmente delle distorsioni che è praticamente impossibile correggere, che provocano un degrado del segnale che si risolve in una sensibile diminuzione della sua qualità; lo stesso problema si riscontra nel caso di una duplicazione anche quando è effettuata con le apparecchiature professionali più sofisticate, e va a sommarsi tutte le volte che si esegue una copia della copia. Al contrario nel caso digitale, in cui il segnale è composto da una serie di numeri, è

estremamente più semplice evitare e correggere tali eventuali errori di trasmissione evitando il degrado del segnale; per lo stesso motivo è estremamente più semplice ottenere una copia del video digitale identica all'originale, anche in seguito ad una lunga sequenza di duplicazioni.

- Maggiore durata: i supporti su cui sono memorizzati i segnali analogici tendono a subire un degrado nel tempo ed ad ogni utilizzo, al contrario il formato digitale può essere mantenuto per lunghi periodi di tempo senza subire modifiche con maggiore semplicità.
- Maggiore possibilità di manipolazione: un segnale analogico una volta memorizzato è difficile da manipolare e modificare, sia perché la modalità di accesso sequenziale propria del formato analogico, sia per la necessità di strumenti professionali adatti per compiere anche le azioni più semplici. Al contrario in ambito digitale grazie alla possibilità di un accesso casuale ai dati ed al fatto che si hanno a disposizione numerosi strumenti anche amatoriali che permettono di manipolare un video con maggiore facilità.

In questo capitolo esporremo alcuni concetti basilari riguardanti la rappresentazione del video e dell'audio, così da fornire alcune basi teoriche per comprendere al meglio i concetti di compressione e conversione video trattati nei capitoli successivi.

1.1 Cattura Video

Il video (dal latino *Io vedo*) è la codifica sotto forma di un segnale elettronico, sia esso analogico o digitale, di una sequenza di immagini che rappresentano scene in movimento, la cui nascita ci fu all'inizio del secolo scorso con l'avvento delle prime trasmissioni televisive. Un'immagine video viene creata registrando una scena reale tramite una videocamera, questo strumento cattura le immagini che che si trova davanti proiettandole su una superficie sensibile al suo interno, in maniera molto simile al funzionamento di una macchina fotografica, con la differenza che, piuttosto che impressionare una pellicola fotosensibile, l'immagine in questione viene scansionata orizzontalmente da un sensore che si muove lungo l'immagine abbastanza velocemente da permetterne la cattura prima che la stessa si muova troppo. Nel caso analogico tale sensore legge l'immagine percependo l'intensità luminosa in ogni punto e codificandola in modo continuo in un segnale di uscita le cui fluttuazioni di ampiezza rispecchiano in modo direttamente proporzionale le variazioni luminose, trasformando così un immagine bidimensionale in un

segnale elettronico monodimensionale; alla fine di ogni linea viene poi inserito un impulso di sincronizzazione che indica la fine di una riga e l'inizio della successiva, ed ulteriori impulsi si inseriscono alla fine dell'immagine completa per sincronizzarsi con la successiva, che viene catturata nello stesso modo. Nel caso digitale il procedimento rimane lo stesso ma si ha una diversa codifica; la scena naturale, continua in tutte le sue dimensioni, viene campionata prima temporalmente, creando una serie di immagini immobili; in seguito ciascuna di esse viene campionata spazialmente tipicamente suddividendo l'immagine tramite una griglia a cui ciascun singolo elemento, una piccola parte della superficie dell'immagine, viene associato un singolo valore numerico che rappresenta l'intensità luminosa mediata lungo tutta la sua superficie.

1.2 Basi del video

Per cercare di capire come un segnale video viene effettivamente visualizzato faremo riferimento alla tecnologia di base, ancor oggi utilizzata sebbene ormai antiquata, la televisione con il tubo a raggi catodici; che è composta essenzialmente di tre componenti principali:

I. ***Il Catodo:*** Ad un estremità del tubo catodico, al cui interno viene creato il vuoto, è presente questo strumento elettronico composto da un elemento metallico che, una volta riscaldato all'incandescenza emette elettroni, che vengono in seguito focalizzati in un fascio coerente (chiamato appunto raggio catodico o pennello elettronico), che viene diretto verso l'anodo, situato in prossimità dello schermo.

II. ***Giogo di deflessione:*** Insieme di elettromagneti che crea un campo magnetico utilizzato per focalizzare e direzionare il fascio di elettroni in modo che colpisca un punto preciso dello schermo; grazie al campo magnetico il fascio di elettroni viene fatto scorrere lungo tutta la superficie dello schermo una linea alla volta secondo la stessa modalità utilizzata per la cattura dell'immagine. Appositi circuiti elettronici pilotano gli elettromagneti del giogo in modo da effettuare una scansione in perfetta sincronia.

III. ***Un monitor:*** Schermo di vetro o di materiale plastico la cui superficie è scandita secondo una matrice predefinita di righe successive, chiamata raster, e l'immagine è creata modulando l'intensità del fascio elettronico secondo l'andamento del segnale video ed il cui lato interno è coperto di fosfori, materiali fosforescenti che si illuminano una volta colpiti dagli elettroni, permettendo

all'utente dall'altro lato dello schermo di vedere l'immagine, la persistenza della luminosità dei fosfori fa sì che lo schermo rimanga impresso abbastanza a lungo da permettere al nostro cervello di percepire l'immagine nella sua interezza.
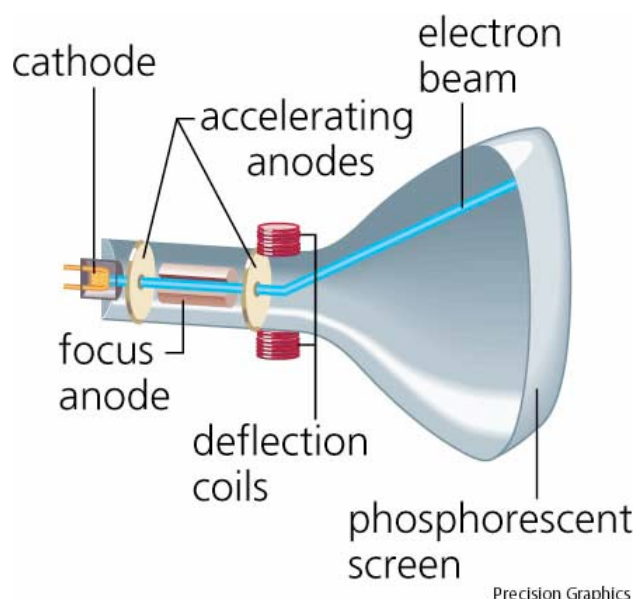
Immagine 1.1 - Tubo a raggi catodici

In questo modo il segnale viene nuovamente convertito in una sequenza di immagini che, ad un frequenza adeguata, verranno percepite dall'occhio e dal cervello umano come una sequenza continua ed in movimento. Il segnale video dispone di un certo numero di caratteristiche che ne individuano le specifiche tecniche, ne permettono il confronto e ne danno un giudizio sulla qualità.

1.2.1 Caratteristiche di un video

Le caratteristiche principali con cui viene analizzato un video sono le seguenti:

- ***Risoluzione video:*** la risoluzione video è il parametro che indica la quantità di informazione è utilizzata per memorizzare ciascuna immagine della sequenza temporale, e di conseguenza è un indice del grado di qualità e del numero di dettagli della sua rappresentazione della stessa. Si divide in risoluzione verticale e risoluzione orizzontale ed è calcolata e specificata diversamente a seconda che ci troviamo in ambito analogico oppure in ambito digitale.
    - *Analogico:* in questo caso il parametro più importante e spesso l'unico ad essere indicato è la risoluzione verticale, essa rappresenta il numero di linee utilizzate per scandire un'immagine, ciascuna delle quali riporta un segnale continuo le cui variazioni corrispondono direttamente alle variazioni luminose; un maggior numero di linee corrisponde ad una maggiore quantità di scan-

sioni. La risoluzione orizzontale invece viene calcolata diversamente, sebbene abbia un legame con la risoluzione verticale, dato che questo parametro riguarda direttamente la dimensione del sensore di scannerizzazione (ed una più elevato numero di linee comporta necessariamente la riduzione delle dimensioni del sensore) poiché tale parametro influenza direttamente la quantità di dettagli che è possibile riprodurre; per capire ciò basti pensare ad una serie di linee verticali poste a distanza ravvicinata l'una dall'altra, se l'area del sensore è superiore allo spazio tra le linee esse verranno visualizzate come una superficie continua. La risoluzione orizzontale è espressa come una frequenza, calcolando il numero massimo di linee verticali bianche e nere intervallate che possono essere rappresentate correttamente lungo una linea orizzontale; tale scansione si traduce in un segnale con la massima variazione nel tempo, quindi massima frequenza (il nero rappresenta la minima intensità luminosa e si traduce in un segnale nullo, mentre il bianco in un segnale al suo massimo)

- *Digitale:* Nel caso digitale c'è una maggiore uniformità nel concetto di risoluzione orizzontale o verticale, in tale ambito le immagini sono visualizzate come una griglia ortogonale di aree uniformi chiamata *raster*, tali aree uniformi sono chiamate *pixel* (contrazione di *picture element*) e rappresentano una piccola porzione della superficie dell'immagine a cui è associato un singolo valore di luminosità. La risoluzione in questo caso indica il numero di pixel che compongono un immagine, aumentare questo numero significa diminuire la superficie assegnata a ciascun pixel e di conseguenza aumentare il dettaglio dell'immagine (se ad esempio si quadruplica il numero di pixel una superficie a cui precedentemente era assegnato un solo valore ora viene suddivisa in quattro superfici diverse, ottenendo così quattro valori di luminosità diversi). La risoluzione si calcola suddividendola in due valori, il numero di aree uniformi che compongono in senso orizzontale il raster di un'immagine video è chiamato risoluzione orizzontale, mentre il numero di aree uniformi che compongono in senso verticale il raster di un'immagine video risoluzione verticale; la notazione con cui viene espressa secondo la seguente notazione: AxB dove A rappresenta la risoluzione orizzontale e B quella verticale, il numero di pixel complessivo è il prodotto tra questi due valori. Un altro parametro delle immagini digitali che spesso accompagna la risoluzione è la *pro-*

fondità o il numero di *bit per pixel* (*bpp*) che sta ad indicare il numero di bit con cui viene rappresentato il valore di luminosità associato a ciascun pixel, più è ampio tale valore maggiore il numero di gradazioni esprimibili nella rappresentazione dell'immagine.

- ***Aspect Ratio:*** l'aspect ratio, o *rapporto d'aspetto*, è la proporzione tra la larghezza e l'altezza dell'immagine video, quindi delle singole immagini che lo compongono; tipicamente espresso sotto forma di proporzione secondo la notazione X:Y indica che l'altezza dell'immagine è X/Y per la larghezza. Al giorno d'oggi i rapporti più comuni sono il 4:3 (corrispondente ad un rapporto di 1,33) chiamato anche *fullscreen*, utilizzato soprattutto nell'ambito televisivo, ed il 16:9 (corrispondente a 1,78) o *widescreen*. Nel caso di video digitale il rapporto d'aspetto può essere indicato non solo per l'intera immagine ma anche per i singoli pixel che spesso hanno forma rettangolare piuttosto che quadrata; tale indice è chiamato *pixel aspect ratio*.
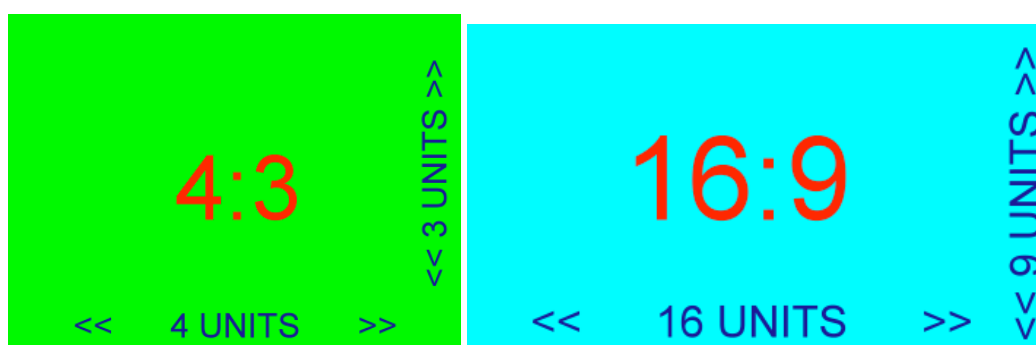
Immagine 1.2 - confronto tra i rapporti di aspetto

- ***Frame Rate:*** indica il numero di immagini in sequenza contenute nell'unità di tempo, e si calcola in immagini al secondo o *frame per second* (*fps*). Perché una sequenza di immagini sia percepita continua al nostro occhio è stato calcolato necessario un minimo di 15 fps ma ottenendo una bassa qualità ed un immagine non perfettamente fluida; per ottenere un effetto di movimento fluido è necessario un frame rate di almeno 24 fps (lo stesso utilizzato nelle vecchie pellicole cinematografiche), che consente una buona qualità nella maggior parte delle situazioni, ma non risolve il problema del *ficker* o *traballamento* che consiste in periodiche fluttuazioni della luminosità dello schermo dovute alla degradazione dei fosfori. Per ovviare a questo problema bisogna far si che avvenga una riaccensioni dello schermo (*refresh screen*) ad una frequenza adeguata, cioè almeno 50 fps.

- ***Scansione delle Immagini:*** La scansione delle immagini è una proprietà del video che indica in che modo avviene la scansione delle immagini nella visualizzazione; esistono due tipi di scansione utilizzati attualmente:
  - *Scansione Interlacciata:* chiamata anche *interlacing* o *interlacciamento*, è il metodo utilizzato per primo nelle trasmissioni televisive e serve a ridurre la banda necessaria alla visualizzazione delle immagini alle frequenze necessarie per ottenere un'immagine fluida e priva di traballamento; il problema deriva da fatto che in quel periodo la tecnologia per effettuare una scansione a tutto lo schermo che era composto di oltre 500 linee ad una frequenza di 50 volte al secondo erano proibitive e quindi non adatte al mercato a cui si faceva riferimento. Tale scansione viene effettuata suddividendo ciascun frame in due campi (o *fields*) ciascuno dei quali contiene la metà dell'informazione complessiva dell'immagine; tale suddivisione viene eseguita numerando le linee di scansione del frame e assegnando al primo campo esclusivamente le linee dispari della scansione mentre il secondo conterrà le linee pari, la visualizzazione dell'immagine completa avviene in due passate. In questo modo si riesce ad ovviare a tutti i problemi, il segnale video veniva generato con un frame rate effettivo più basso (almeno 25 fps abbastanza per avere una sensazione di fluidità nell'immagine) ma il refresh dello schermo avveniva in effetti ad una frequenza doppia, riuscendo così a prevenire il ficker, ciò grazia al fatto che veniva scandita solo la metà delle linee effettive dello schermo.
  - *Scansione Progressiva:* A differenza dell'interlacing la scansione progressiva (o *progressive scanning*) non suddivide il frame intero in campi, ma lo scannerizza direttamente una linea dopo l'altra. Questa modalità di scansione è nata con l'avvento dei primi computer, quando ormai la tecnologia era giunta al punto da permettere tale soluzione, inoltre la scansione interlacciata non funzionava molto bene sugli schermi dei Computer a causa del loro maggiore contrasto e luminosità. La scansione progressiva ha il vantaggio di mantenere un'immagine di qualità più elevata, una maggiore risoluzione, ed elimina alcuni artefatti soprattutto dovuti ad effetti di movimento apparente percepiti come una parziale vibrazione in un'immagine ferma o con particolari combinazioni di colori e linee in finissima successione (effetto moiré). Per mantenere la compatibilità con i segnali interlacciati ancora molto utilizzati (per trasmissioni analogiche o satellitari e DVD) nei nuovi apparecchi a scansione

progressiva viene utilizzata una procedura di rimozione dell'interlacciamento (*deinterlacing*) ma essa non può comunque ottenere una qualità video uguale a quella che offrono gli apparecchi a scansione progressiva in quanto la quantità di dati che il segnale contiene è pressoché la metà.

- ***Spazio dei Colori:*** Finora abbiamo parlato in generale di un singolo segnale di luminosità riguardo alla memorizzazione e alla rappresentazione di un video, il che corrisponde ad un immagine monocromatica; per rappresentare invece un immagine a colori si richiede generalmente l'uso di tre valori per ogni singolo campione ed il metodo scelto per rappresentare la luminosità (*brightness* o *luminance*) ed il colore definisce lo spazio dei colori. Uno spazio dei colori è la combinazione di un modello di colore, cioè un modello matematico astratto che descrive un modo per rappresentare i colori come combinazioni di numeri, tipicamente come tre o quattro valori detti componenti colore che definiscono il sistema di coordinate spaziali, e di una appropriata funzione di mappatura che esprime la regole specifiche necessarie all'implementazione del modello astratto per l'uso voluto. Esistono vari spazi dei colori che possono essere utilizzati, di seguito esporremo i più comuni nelle applicazioni video.
  - *RGB:* L'RGB (red, green, blue) è un modello additivo in cui ogni campione dell'immagine a colori viene rappresentato da tre valori distinti che indicano rispettivamente le componenti dei tre colori primari, rosso, verde e blu; la somma di queste componenti secondo le giuste proporzioni permette di ottenere quasi tutto lo spettro dei colori visibili ad esempio l'unione dei tre componenti secondo uguali proporzioni permette di ottenere tutta la scala dei grigi, dal nero (tutti i colori al minimo) al bianco (tutti i colori al massimo) la combinazione delle coppie di colori dà il cìano (verde e blu), il magenta (blu e rosso) e il giallo (rosso e verde). La rappresentazione numerica di un colore nel sistema RGB consiste nella rappresentazione di ciascuna componente, generalmente ciascuna di esse rappresentata da un intervallo numerico, i più comuni sono l'intervallo continuo [0,1] usato soprattutto nelle formule, la rappresentazione percentuale, oppure l'intervallo discreto [0, 255], quest'ultimo utilizzato generalmente in computer science dato che corrisponde all'uso di un byte per ogni componente, indicato in valori decimali o esadecimali. Più specificamente i tre colori principali individuano forme d'onda (radiazioni luminose) di periodo fissato. A una lunghezza d'onda di 700 nm corrisponde il

rosso, a 546.1 nm il verde, a 435.8 nm il blu, corrispondenti alle lunghezze d'onda che stimolano maggiormente l'occhio umano. Il modello RGB è molto utilizzato ancora oggi soprattutto nella tecnologia dei display e in alcune codifiche di immagini ed è utilizzato come base per molti altri sistemi.

Immagine 1.3 - Divisione dell'immagine nel formato RGB

- *HSV:* l'HSV (*Hue, Saturation, Value* o *tonalità, saturazione e luminosità*) modello utilizzato soprattutto in computer grafica orientato alla prospettiva umana, essendo basato sulla percezione che si ha di un colore in termini di tinta, sfumatura e tono. Il sistema di coordinate è cilindrico è il modello HSB è definito come un cono. La *tonalità* (H) o *tinta* in teoria dei colori, indica un colore "puro", ovvero caratterizzato da una singola lunghezza d'onda all'interno dello spettro visibile (o spettro ottico) della luce e viene misurata da un angolo intorno all'asse verticale, con il rosso a 0 gradi, il verde a 120 e il blu a 240. L'altezza del cono rappresenta la luminosità (V) è la quantità totale di luce che una sorgente luminosa appare emettere (o che appare riflessa da una superficie); con lo zero che rappresenta il nero e l'uno il bianco. Infine la *saturazione* (S) o *purezza* è l'intensità di una specifica tonalità, una tinta molto satura ha un colore vivido e squillante, al diminuire della saturazione, il colore diventa più debole e tende al grigio; essa viene invece misurata da zero, sull'asse del cono, a uno sulla superficie del cono.
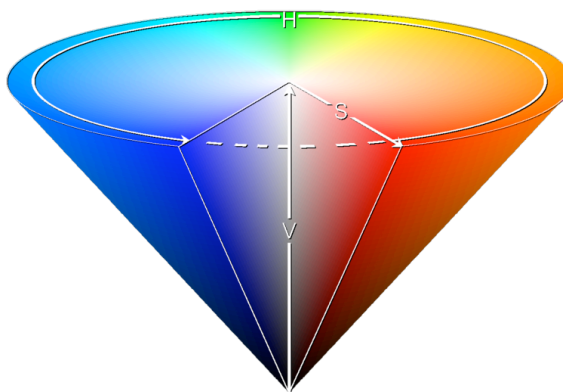
Immagine 1.4 - Spazio del colore HSV

- *YCrCb:* L'YCrCb è una famiglia di spazi dei colori molto utilizzati nelle applicazioni video e negli standard televisivi; non è un modello assoluto ma un particolare modo di rappresentare più efficientemente il colore RGB. Y rappresenta i valori di luminosità relativi all'immagine acromatica; le altre due componenti rappresentano la *crominanza* (*chrominance* o *chroma*) cioè l'informazione relativa al colore, i due componenti rappresentano la crominanza rossa (Cr) e la crominanza blu (Cb) calcolati come la differenza tra il valore del rispettivo colore e la luminanza Y12; sebbene si possa estrarre anche il valore della crominanza relativa al verde esso non è necessario in quanto può essere ricavato dagli altri tre valori. Questo modello permette di ottimizzare la rappresentazione dell'informazione e ridurre l'ampiezza di banda del segnale video tramite il processo che prende il nome di *chroma subsampling* (*sottocampionamento della crominanza*), una pratica che corrisponde nell'utilizzare una maggiore risoluzione per memorizzare il componente Y ed una risoluzione inferiore per i due valori di crominanza, ciò è possibile grazie al fatto che il nostro sistema visivo è meno sensibile alle informazioni cromatiche rispetto alla luminosità. Esistono vari schemi di sottocampionamento, di seguito mostreremo i più utilizzati negli standard MPEG; tali schemi sono definiti secondo la seguente convenzione 4:n:m, il 4 rappresenta la frequenza di campionamento dei valori di luminosità di 13,5MHz (la frequenza standard) mentre gli altri due numeri indicano i fattori di campionamento dei valori Cb e Cr lungo la risoluzione orizzontale rispetto al primo valore.
- 4:4:4 - Questo schema usa la stessa risoluzione per la luminanza e per la crominanza, non avendo alcun guadagno rispetto all'RGB classico è utilizzato soprattutto nel caso di video ad alta definizione (in alcuni casi questo simbolo è utilizzato proprio per indicare l'RGB).
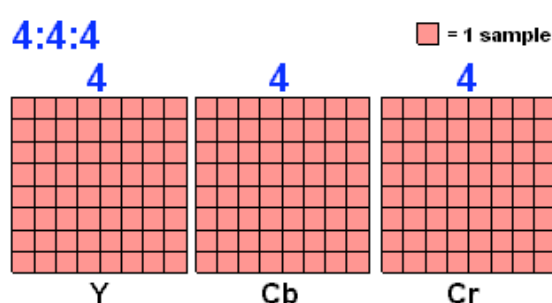
Immagine 1.5 - Schema di sottocampionamento 4:4:4

- 4:2:2 - In questo schema i valori di crominanza hanno la stessa risoluzione verticale ma metà della risoluzione orizzontale rispetto alla luminanza; in pratica per ogni linea dell'immagine vengono catturati un campione di crominanza per ogni due campioni di luma. In questo caso si ottiene una diminuzione della quantità d'informazione di un terzo.
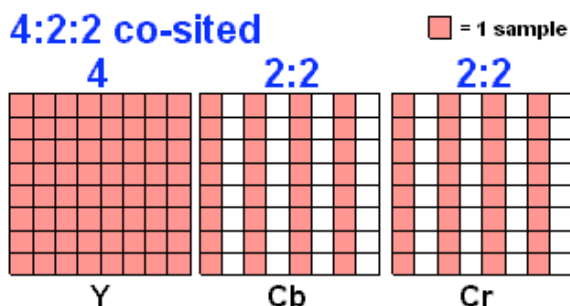
Immagine 1.6 - Schema di sottocampionamento 4:2:2

- 4:2:0 - Il metodo maggiormente utilizzato dato che riduce la quantità dell'informazione della metà. In questo caso i valori di crominanza hanno la metà della risoluzione sia in orizzontale che in verticale, dunque ogni gruppo di quattro campioni di luminanza si cattura un solo campione di crominanza.
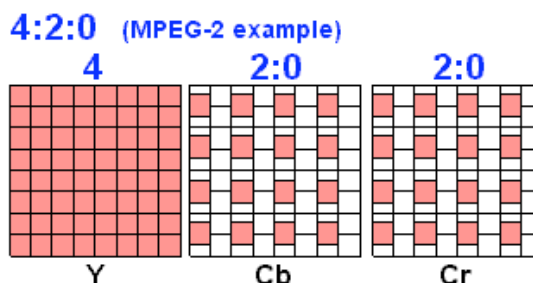
Immagine 1.7 - Schema di sottocampionamento 4:2:0

Le equazioni di conversione dal sistema RGB al YCrCb:

$$Y = k_r * R + k_b * B + (1 - k_r - k_b) * G$$

$$C_r = \frac{0,5}{1 - k_r} * (R - Y)$$

$$C_b = \frac{0,5}{1 - k_b} * (B - Y)$$

E le inverse:

$$R = Y + (\frac{1 - k_r}{0,5}) * C_r$$

$$G = Y - (2 * \frac{k_r * (1 - k_r)}{1 - k_r - k_b}) * C_r - (2 * \frac{k_b * (1 - k_b)}{1 - k_r - k_b}) * C_b$$

$$B = Y + (\frac{1 - k_b}{0,5}) * C_b$$

- Oltre le caratteristiche elencate nei video sono definite anche altre proprietà, in particolare nei video digitali si definiscono il Bit rate, un indicazione della frequenza dell'informazione trasmessa dal video, misurata in bit per secondo (*bps*), un maggiore bit rate corrisponde ad una migliore qualità. Un'altra caratteristica dei video digitali è il metodo di compressione, utilizzato per ridurre la dimensione del video, generalmente estremamente elevata.

1.2.2 Standard video

Di seguito elencheremo alcuni degli standard video più utilizzati attualmente, per le trasmissioni televisive e la codifica dei DVD-video

- ***NTSC:*** L'NTSC è uno standard per la creazione, trasmissione e ricezione di contenuti video per le aree geografiche Corea, Giappone, Canada, USA, Messico e alcuni paesi sudamericani; il suo nome è la sigla di National Television System(s) Committee, l'ente di standardizzazione industriale che lo ha creato. L'NTSC può essere di tipo interlacciato (60i) o progressivo (30p), nel caso specifico del 60i abbiamo 60 semiquadri che compongono 30 fotogrammi al secondo (in realtà la frequenza esatta è di 29,97 al secondo - 60 Hz); prevede l'utilizzo di 525 linee per la definizione di un fotogramma completo (262.5 linee per ogni campo necessario all'interlacciamento) di cui solo 486 vengono realmente utilizzate per comporre l'immagine, mentre le restanti vengono impiegate per altre informazioni come il sync, il captioning e la velocità di tracciamento. Il formato 60i è il più adatto (e diffuso) in ambito televisivo e di diffusione/creazione di materiali per la fruizione via CRT. Nel caso del 30p abbiamo una scansione di 30 quadri interi senza alcuna tecnica di interlacciamento. La codifica dei colori sfrutta un sistema basato sulla luminanza-crominanza (della stessa famiglia degli spazi YCbCr), la

cui sottoportante di crominanza ha una banda di 3.579545 MHz. La risoluzione in pixel del fotogramma NTSC dipende dal veicolo di diffusione o creazione, queste le risoluzioni più in uso:

* Ambito Televisivo: 646x486
* NTSC DV(Digital Video): 720x480

- ***PAL:*** Lo standard PAL (*Phase Alternating Line*), utilizzato in gran parte dell'Europa, Asia, Africa e Sud America e in Oceania, creato dal tedesco Walter Bruch alla Telefunken in Germania, usato per la prima volta nel 1963. Il PAL è un formato interlacciato con un frame rate di 50 campi al secondo, che compongono 25 fotogrammi al secondo, ciascuno composto di 625 linee di cui 576 utilizzate per la composizione dell'immagine ed una sottoportante di crominanza di 4.433618 MHz; la risoluzione tipica del formato PAL è di 768x576. Per via della sua maggiore risoluzione lo standard PAL è in genere di qualità maggiore

SISTEMA	NTSC	PAL
Linee/Frame	525/60	625/50
Frequenza (risoluzione) orizzontale	15.734 kHz	15.625 kHz
Frequenza verticale (frame rate)	60 Hz	50 Hz
Frequenza della sottoportante del colore	3.579545 MHz	4.433618 MHz
Larghezza di banda del video	4.2 MHz	5.0 MHz
Portante del suono	4.5 MHz	5.5 MHz

- ***HDTV:*** Lo standard di più recente introduzione, dotato di una risoluzione e di una qualità decisamente superiori rispetto ai vecchi standard analogici.
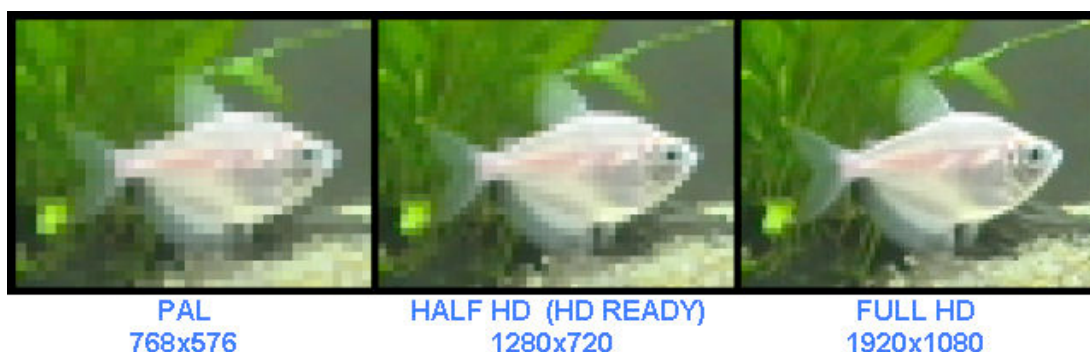
Immagine 1.8 - Confronto della qualità delle immagini

La tecnologia HDTV comprende quattro formati video, che differiscono sia per la risoluzione effettiva che per le modalità di scansione dell'immagine.

- Il formato HR.HDTV (Half Resolution High Definition TV) è di 960x540 ovvero un quarto della definizione FULL HD (vedi sotto) usato da alcune emittenti televisive per trasmissioni TV dato il buon compromesso risoluzione/banda richiesta. Inoltre essendo comunque un formato 16:9 l'upscaling su televisori FULL HD o HD READY avviene senza distorsioni.
- Il formato 720p, comunemente chiamato HD READY (i televisori che lo supportano riportano il logo HD ready, cioè "pronto per l'alta definizione"), presenta una risoluzione complessiva di 921.600 pixel (1280×720) con scansione progressiva, ovvero per ciascun ciclo di trasmissione di un fotogramma (50 o 60 Hz a seconda dei Paesi) viene trasmesso l'intero quadro dell'immagine. Ogni aggiornamento coinvolge tutte le 720 linee e i 921.600 pixel dello schermo.
- Il formato 1080i, comunemente chiamato FULL HD, presenta una risoluzione complessiva di 2.073.600 pixel (1920×1080) con scansione interlacciata, ovvero per ciascun ciclo viene trasmesso un semiquadro formato alternativamente dalle sole linee pari o dispari dell'immagine. Quindi ogni aggiornamento coinvolge 540 righe e 1.036.800 pixel.
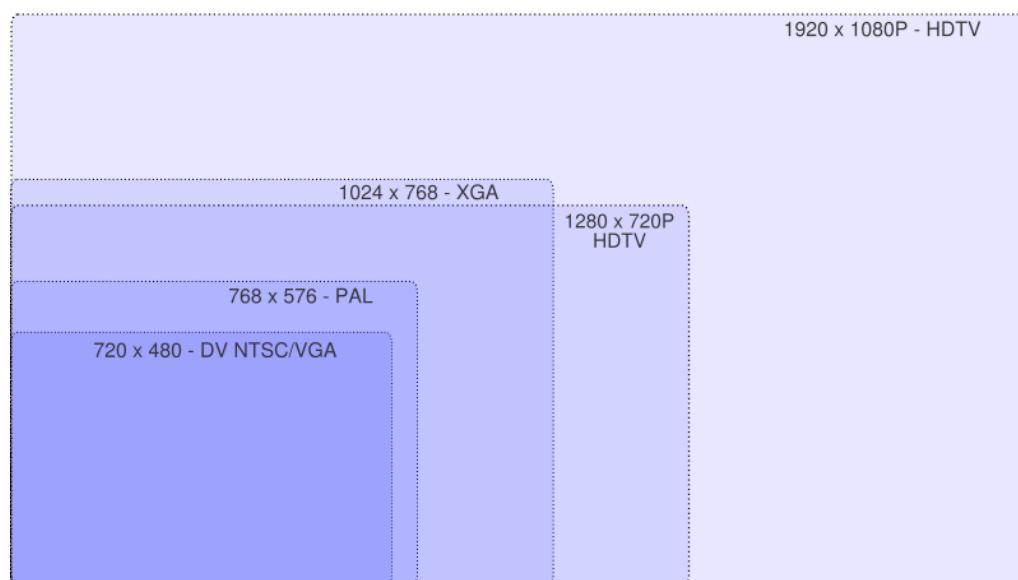
Immagine 1.9 - Confronto tra le risoluzioni dei vari formati

- Il formato 1080p, anch'esso chiamato FULL HD, è il più recente ed equivale alla versione con scansione progressiva del 1080i, per cui ogni aggiornamento coinvolge tutte le 1080 linee e i 2.073.600 di pixel dello schermo.

1.3 Audio

Finora ci siamo concentrati sull'informazione visiva e sulla sua rappresentazione, ma un altrettanto importante elemento dei contenuti multimediali è la componente sonora; così come per il video anche per l'audio c'è stato un passaggio da una rappresentazione analogica ad una digitale nella maggior parte dei campi. Il suono è prodotto dalle vibrazioni della materia che inducono una variazione di pressione nell'aria che le circonda che viene percepita dal nostro apparato uditivo. Il pattern di queste oscillazioni è chiamato *waveform*, ed il suo parametro principale è la frequenza, che indica il numero di oscillazioni al secondo, questo parametro è estremamente importante perché solo una parte delle frequenze possibili in un onda sonora sono effettivamente rilevabili dall'orecchio umano, secondo la seguente tabella:

Suoni udibili	da 20 Hz a 20 kHz
Ultrasuoni	da 20 kHz a 1 GHz
Ipersuoni	da 1 GHz a 10 THz

Un secondo parametro è l'ampiezza, che misura il displacement della pressione dell'aria rispetto al suo stato quiescente.

1.3.1 Rappresentazione digitale del Suono

Il suono nel mondo analogico è continuo sia nel tempo che in ampiezza, che può essere misurata, con un arbitrario grado di accuratezza, in qualsiasi istante di tempo; un suono digitale al contrario è definito solo in alcuni punti nel tempo, ed il segnale può avere sono un limitato numero di valori. Per trasformare il segnale sonoro da analogico in digitale si suddivide in due fasi distinte:

I. ***Campionamento:*** il campionamento consiste nell'esaminare il segnale ad un preciso punto del tempo, secondo degli intervalli ben precisi ad un frequenza adeguata (frequanza di campionamento o sampling rate); tale frequenza viene decisa applicando il teorema del campionamento (o teorema di Nyquist-Shannon), il quale afferma che per campionare un segnale senza perdite è necessaria una frequenza di campionamento almeno doppia rispetto alla massima frequenza del segnale stesso. tipicamente le frequenze di campionamento utilizzate nell'audio digitale sono 44.1 kHz o 48 kHz.

II. ***Quantizzazione:*** il processo di quantizzazione consiste nel trasformare i segnali ottenuti dal campionamento, che sono espressi da un insieme di valori

continuo, in un insieme di valori finito e discreto. Per ottenere questo si divide il range dei valori in una serie di livelli pari a 2^n dove n è il numero di bit con cui si vuole codificare il segnale, ed ogni campione si assegna il valore discreto corrispondente al livello più vicino al suo valore.
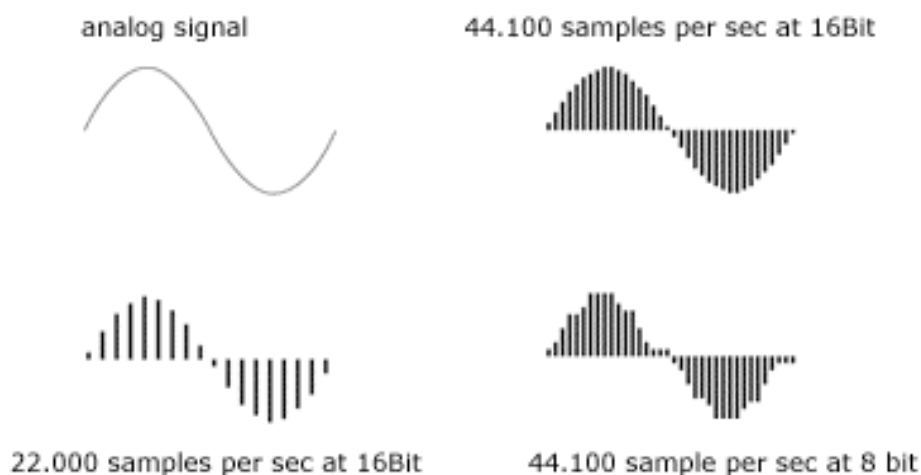
immagine 1.10 - esempio di campionamento e quantizzazione.

1.3.2 Psicoacustica

La psicoacustica è lo studio della percezione soggettiva umana dei suoni; è applicata oggi in molti campi: dall'ingegneria informatica all'ingegneria acustica, nella musica, dove i musicisti e gli artisti continuano a creare nuove sensazioni acustiche, rompendo la tradizionale percezione della sonorità reale. Il modello psicoacustico conferisce qualità alla compressione audio di tipo "lossy" indicando quale parte del segnale audio da comprimere può essere rimossa o pesantemente compressa senza generare problemi, cioè senza perdite significative nella qualità del suono. La psicoacustica è fortemente basata sull'anatomia umana, soprattutto sulle limitazioni di percezione dell'orecchio. Dette limitazioni, per riassumere, sono:

- Limiti alle alte frequenze
- Soglia assoluta di udibilità
- Soglia del dolore
- Temporal masking (Mascheramento temporale)
- Simultaneous masking (Mascheramento simultaneo)

1.3.1.1 Frequenza

L'orecchio umano può udire i suoni nell'intervallo dai 20 Hz ai 20.000 Hz, questo limite superiore tende ad abbassarsi con l'avanzare degli anni, molti adulti non sono in grado di udire frequenze oltre i 16 kHz. Nel tratto centrale dell'intervallo delle

frequenze udibili l'orecchio ha una risoluzione di circa 2 Hz, il che significa che sono percepibili modifiche di un suono in altezza solo se queste sono maggiori di 2 Hz, tuttavia differenze d'altezza minori sono percepibili in altre maniere, ad esempio l'interferenza tra due altezze può essere spesso ascoltata come una differenza d'altezza di bassa frequenza; questo effetto di variazione di fase sul suono risultante è comunemente conosciuto come fenomeno dei battimenti. L'effetto della frequenza sull'orecchio umano segue una base logaritmica, in altre parole, la maniera in cui l'altezza di un suono che viene percepita è funzione esponenziale della frequenza.

1.3.1.2 Intensità

Se si prende in considerazione l'intensità del suono, l'intervallo udibile è enorme: il limite inferiore è definito a 0 dB, mentre il limite superiore attualmente non è fissato. È possibile individuare un limite superiore approssimativo considerando il punto in cui l'intensità del suono è tale da danneggiare l'orecchio. In questo modo il limite dipende dalla durata del suono, poiché l'orecchio può sopportare 120 dB per un breve periodo, ma può subire sordità permanente se esposto per lungo tempo a un suono oltre gli 80 dB. Una più rigorosa esemplificazione dei limiti minimi di udibilità determina che la soglia minima dove un suono può essere sentito dipende dalla sua frequenza. Misurando questa intensità minima utilizzando toni di test a varie frequenze, possiamo ricavare una curva di "Frequenza Dipendente" detta Soglia Assoluta della percezione Sonora o *Absolute Threshold of Hearing* (ATH).
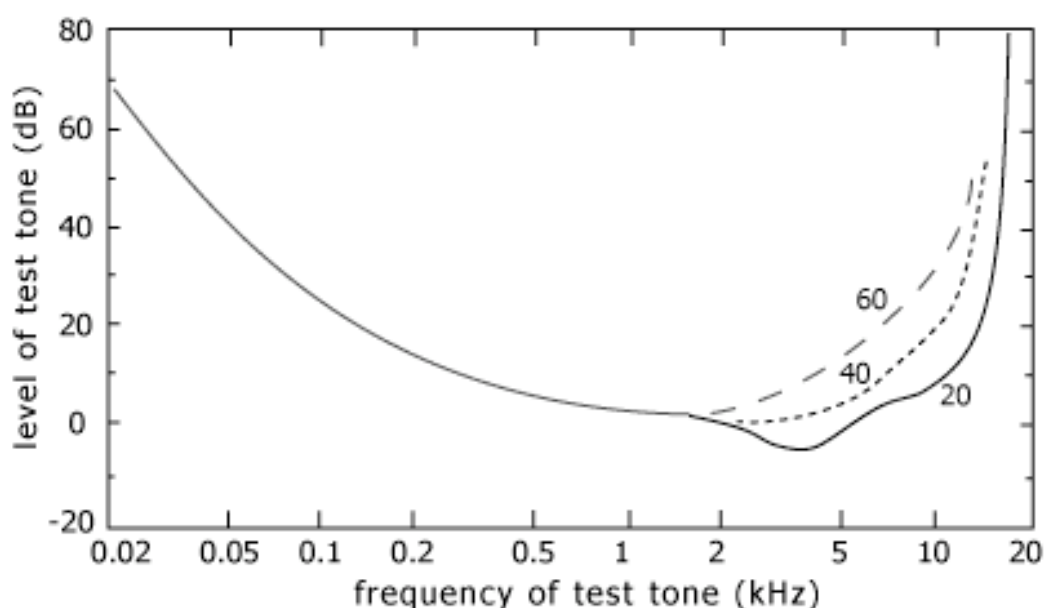
Immagine 1.11 - ATH di un uomo al variare dell'età

L'ATH è costituito dalla più bassa tra le *curve di pari intensità sonora* o *equal-loudness contour*; le curve di pari intensità indicano il livello di pressione acustica (in dB), distribuite su un intervallo di frequenze udibili, che sono percepite allo stesso livello di volume sonoro.

1.3.1.3 Effetti di mascheramento

In alcune situazioni, un suono normalmente udibile può essere mascherato da un altro suono. Ad esempio, la conversazione a una fermata di autobus può diventare completamente impossibile se si sta avvicinando un rumoroso autobus. Questo fenomeno è chiamato "mascheramento". Un suono più debole è detto "mascherato" se è reso inaudibile dalla presenza di un suono più forte. Se due suoni vengono prodotti simultaneamente e uno è mascherato dall'altro, si parla di mascheramento simultaneo. Un suono di frequenza prossima a quella del suono più forte è mascherato più facilmente rispetto a uno di frequenza molto diversa. Per questo motivo, il mascheramento simultaneo è anche chiamato "mascheramento di frequenza". La tonalità di un suono è parzialmente determinato da questa abilità di mascherare gli altri suoni. I modelli computerizzati che calcolano il masking causato da suoni deve pertanto classificare la loro vetta individuale all'interno dello spettro accordando la loro tonalità. Allo stesso modo, un suono leggero emesso appena dopo la fine del suono alto è mascherato da quest'ultimo e persino un suono leggero appena prima di un suono alto può essere mascherato da un suono alto, questi due effetti sono chiamati rispettivamente anticipo e ritardo del temporal-masking(mascheramento temporale).

Capitolo 2

Compressione Audio/Video

Nonostante il continuo incremento della banda delle connessioni di rete e l'aumento di dimensioni dei dispositivi di contenimento la dimensione di un video digitale così come viene creato è ancora troppo elevata per essere trattato naturalmente, se non con risultati da qualità estremamente bassa, è perciò necessario utilizzare dei metodi per ridurre la quantità di dati utilizzati per la rappresentazione di un video, ed è a questo che è necessaria la compressione video. Un segnale può essere compresso rimuovendo la ridondanza dell'informazione, in una compressione senza perdita è rimossa esclusivamente la ridondanza statistica in modo che il segnale originale possa essere ricostruito perfettamente, ma allo stato attuale una compressione del genere ha un efficienza estremamente bassa nel caso di riduzione di immagini e video; le tecniche più pratiche si servono dell'utilizzo di algoritmi di compressione con perdita dell'informazione che riescono ad ottenere un forte grado di riduzione della dimensione del segnale al prezzo di avere un segnale decodificato non identico all'originale, in questo caso l'obiettivo principale consiste nel cercare di ottenere un'efficiente compressione minimizzando la distorsione introdotta tra il segnale originale e quello decodificato. La compressione video opera rimuovendo la ridondanza temporale, spaziale e/o nel dominio della frequenza, ad esempio un'immagine in cui è presente uno sfondo monocromatico ci sono numerose aree uniformi che comportano una significante riduzione spaziale, mentre la ridondanza temporale si basa sul fatto che in una serie di frame in sequenza, che alla frequenza utilizzata, due immagini successive sono generalmente sono molto simili tra loro, e piuttosto che come singole immagini possono essere considerate tramite le differenze tra l'una e l'altra, diminuendo così la quantità di dati necessari. In questo capitolo parleremo della compressione di segnali video digitali, partendo dai metodi di base utilizzati nella maggior parte degli algoritmi focalizzandoci in seguito sui metodi di compressione MPEG-2 e MPEG-4, gli standard effettivamente utilizzati.

2.1 Basi della Compressione

La compressione è il processo di compattare i dati per rappresentarli con un minore numero di bit, la compressione video è il processo di compattare o condensare una sequenza video digitale; un video “Raw” o non compresso richiede tipicamente un’ampia larghezza di banda (approssimativamente 216 Mbps per un secondo di video non compresso di qualità pari a quella televisiva) e la compressione è necessaria per la trasmissione e la conservazione del video. La compressione coinvolge una coppia di sistemi complementari, un compressore (encoder) ed un decompressore (decoder), il primo converte i dati delle sorgente in un forma compressa prima di trasmetterli o conservarli, il secondo riutilizza il segnale compresso per riottenere i dati originali; la coppia encoder/decoder è descritta come un solo strumento chiamato codec. La compressione dei dati è raggiunta rimuovendo la ridondanza, cioè le componenti che non sono effettivamente necessarie ad una fedele riproduzione dei dati, molti tipi di dati contengono ridondanza statistica, e possono essere efficacemente ridotti utilizzando compressioni lossless (prive di perdita) così che il dato ricostruito sia una copia perfetta del dato originale; sfortunatamente il grado di compressioni raggiunto con strumenti lossless nel caso di video e immagini è estremamente limitato. Una compressione lossy (con perdita di informazione) sono necessarie per raggiungere una maggiore compressione, al prezzo di ottenere un segnale ricostruito non perfettamente uguale al segnale di partenza, una maggiore compressione è ottenuta al prezzo di una certa perdita di qualità. I sistemi di compressioni lossy sono basati sul principio della rimozione della ridondanza soggettiva, elementi dell'immagine o del video che possono essere rimossi senza influenzare significativamente la percezione qualitativa dell’osservatore. Molti tipi di codificatori ricercano sia la ridondanza spaziale che quella temporale, per raggiungere un più alto grado di compressione, nel dominio del tempo c’è generalmente una più ampia correlazione (similarità) tra i frame di un video che sono stati catturati in istanti di tempo vicini, specialmente se il frame rate è alto; nel dominio spaziale c’è generalmente un alta correlazione tra i pixel vicini tra loro. I compressori MPEG utilizzano un modello basato su blocchi per la compensazione del moto, la trasformazione, la quantizzazione e la codifica entropica.

2.1.1 Modello Temporale

L'obiettivo del modello temporale è esplorare l’intrinseca ridondanza tra le immagini di un video, spesso nelle sequenze video pictures temporalmente adiacenti

hanno un ampio grado di similarità, ad esempio lo sfondo può rimanere invariato con disturbi nel primo piano; in tal caso l'applicazione della classica strategia di compressione differential pulse code modulation (DPMC) può ottenere una sostanziale compressione, con la predizione applicata all'intera immagine video. La tipica applicazione di questa strategia usa l'ultima immagine per predire quella attuale, ed è chiamata *picture differencing*, e consiste nell'ottenere un frame predetto partendo dal frame di riferimento e sottraendovi il frame corrente, l'output di questo processo è un frame residuo (differenziale) il risultato è un'immagine a livelli di grigio in cui il livello di grigio medio rappresenta una differenza nulla, il grigio luminoso una differenza positiva ed il grigio scuro una differenza negativa; più accurata è la predizione, minore è l'energia contenuta nel frame residuo, il problema di questo metodo è che, anche nelle migliori condizioni, contiene ancora una gran quantità di energia, il che consiste in un elevata quantità d'informazione da comprimere; tale frame è poi codificato e inviato ad un decodificatore che ricrea il frame predetto, vi aggiunge il il frame residuo decodificato e riottiene il frame corrente. Il frame predetto è creato da uno o più frame successivi o precedenti (frame di riferimento). Si può comunque andare oltre notando che quando ci sono dei cambiamenti nell'immagine corrente, spesso sono causati dal movimento di oggetti; un altra occorrenza comune è il movimento dell'intera scena dovuto alla telecamera; ed è proprio il movimento è il responsabile della maggior parte dell'energia presente nel frame residuo, queste osservazioni motivano l'inclusione di un modello del movimento nella predizione per ridurre la quantità di informazione spaziale che deve essere codificata come residuo. La predizione temporale con la compensazione del moto non è universalmente efficace in ogni caso, per esempio un cambio di scena comporta una discontinuità tra le immagini che non può essere predetta da alcun tipo di tecnica, inoltre il modello deve essere adattato al particolare tipo di cambiamento di scena, ad esempio un modello basato sulla traslazione è meno efficace in caso di rotazioni o ingrandimenti.

Immagine 2.1 - Differenza tra due frame

2.1.1.1 Cambiamenti dovuti al moto

I cambiamenti tra i frame possono essere causati dal moto degli oggetti o da quello della telecamera, dalla scoperta di regioni (ad esempio una parte dello sfondo che appare in seguito al movimento di un oggetto in primo piano) o dal cambio d'illuminazione, nel caso dei primi due modi (i più comuni), la differenza si traduce in uno spostamento dei pixel delle scena tra i frame; è dunque possibile stimare la traiettoria di ciascun pixel tra le immagini come *l'optical flow* (Optical Flow è un concetto che considera il moto di un oggetto all'interno di una rappresentazione visuale digitale, tipicamente il moto è rappresentato come un vettore che si origina o termina su un pixel ad esempio in una sequenza di frames; lo scopo dell'Optical Flow è quello di assegnare ad ogni pixel appartenente al frame corrente un motion vector che punta verso la posizione dello stesso pixel in un frame di riferimento successivo). Se l'optical flow è abbastanza accurato è possibile formare una predizione della maggior parte dei pixel del frame corrente muovendo ciascuno di essi dal riferimento lungo il suo corrispondente vettore del moto. In ogni caso non è un sistema pratico di compensazione del moto per varie ragioni, prima tra tutte l'elevata complessità computazionale di una predizione accurata (la più accurata consiste in un vettore per ogni pixel) inoltre l'elevato numero di vettori del moto risultante diminuisce l'efficacia della compressione.
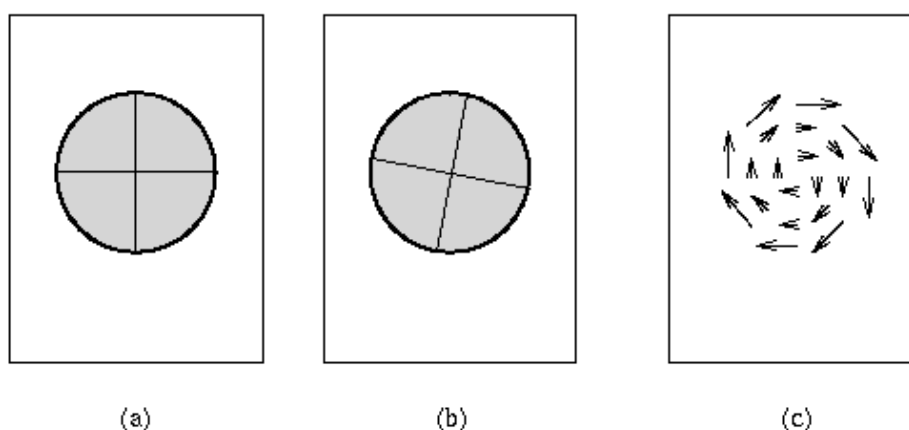
Immagine 2.2 - Optical Flow: tempo t1(a), tempo t2(b), Optical flow(c)

2.1.1.2 Stima basata su Blocchi

Un modo pratico e largamente utilizzato per ottenere una buona compensazione del moto è suddividere ogni immagine in blocchi di MxN pixel ed associare un vettore del moto a ciascun blocco. Per ottenere la predizione si effettua la seguente procedura per ogni blocco:

1. Cercare in un area del frame di riferimento per trovare un blocco MxN di matching; ciò è ottenuto comparando nel frame corrente con alcuni o tutti i possibili blocchi del frame di riferimento e trovare tra essi la regione che fornisca il best-match. Un popolare criterio di matching è la minimizzazione dell'energia di nel blocco residuo formato dalla differenza tra i blocchi comparati; questo processo è conosciuto con il nome di stima del moto (motion estimation).
2. La regione scelta viene poi sottratta dalla regione corrente per ottenere il blocco residuo (motion compensation).
3. Viene calcolato l'offset tra le due regioni (motion vector), che viene trasmesso assieme al blocco residuo creato al passo precedente.

La compensazione del moto basata su blocchi è popolare per un certo numero di ragioni, è relativamente diretta e computazionalmente trattabile, si associa bene con i frame video rettangolari e con i le trasformazioni delle immagini basate su blocchi (ad esempio la Discrete Cosine Transform) e provvede ad un ragionevolmente efficiente modello temporale per molte sequenze video. Ci sono tuttavia alcuni svantaggi, ad esempio gli oggetti reali non hanno bordi netti che si raccordano con dei confini rettangolari, spesso gli oggetti si muovono di un numero di pixel frazionali, inoltre molti tipi di movimento non sono facilmente descrivibili attraverso questo metodo (oggetti deformabili, ingrandimenti, e moti complessi come nuvole), nonostante ciò questo modello è molto utilizzato nella maggior parte degli standard video.
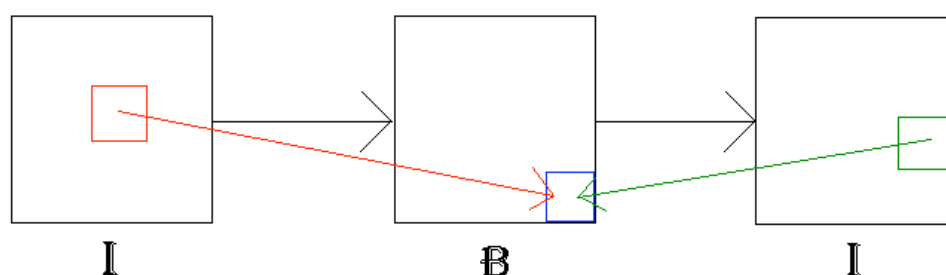
Immagine 2.3 - Compensazione del moto basata su blocchi

2.1.1.3 Dimensioni dei blocchi

Uno dei parametri fondamentali nella predizione del moto è la dimensione delle regioni, una regione di dimensione maggiore produce una minore qualità della compensazione del moto rispetto ad una regione più piccola, ma in compenso ne diminuisce la complessità computazionale ed aumenta il numero di vettori del mo-

to necessari alla descrizione del modello, è necessario trovare il miglio compromesso tra la qualità e la complessità, e nella maggior parte degli standard esso è fissato ad una regione di 16x16 pixel.

2.1.1.4 Sub-Pixel Motion-Compensation

In alcuni casi è possibile migliorare l'accuratezza della predizione utilizzando dei campioni interpolati nel frame di riferimento, questo metodo consiste nel creare dei sottocampioni virtuali sulla base di quelli reali, di fatto aumentando la risoluzione del frame, e permettendo di trovare una corrispondenza maggiore con il frame desiderato rispetto alla campionatura normale; la stima basata su sub pixel coinvolge la ricerca sia dei campioni ai valori interi che nei sottocampioni interpolati nella ricerca del best match; il processo viene eseguito in più passi, nella prima fase si trova il migliore candidato nella griglia dei campioni interi, una volta trovato viene effettuata una seconda ricerca nei mezzi pixel immediatamente prossime al match trovato, se necessario il processo può essere iterato andando a cercare il quarter-pixel prossimo al mezzo pixel ottimo. Arrivare fino alla stima al quarto di pixel migliora significativamente le prestazioni del preditore, utilizzare ulteriori passi generalmente non aumenta ulteriormente il guadagno se non di un fattore minimo.

2.1.2 Modello spaziale

Un immagine video naturale consiste in una griglia di campioni, esse sono spesso difficili da comprimere nella loro forma originale a causa della forte correlazione tra due campioni vicini della stessa immagine, che diminuisce con estrema lentezza al variare della distanza tra campioni; in questo senso è utile il processo di compensazione del moto, dato che le immagini residue che sono il risultato del modello temporale hanno una minore correlazione interna all'aumentare della distanza tra campioni. Lo scopo del modello spaziale (o modello dell'immagine) è quello di scorrelare le immagini residue e convertire ulteriormente i dati in una forma che possa essere compresa efficacemente attraverso l'uso di codificatori entropici; un modello tipicamente è composto di tre componenti principali, la trasformazione che scorrela e compatta i dati, la quantizzazione che riduce la precisione dei dati trasformati ed il riordino che distribuisce i dati in gruppi dai valori significativi.

2.1.2.1 Predictive Image Coding

La compensazione del moto è un esempio di codifica predittiva in cui un codificatore crea una predizione delle regioni del frame corrente basato su un frame precedente e le sottrae al frame corrente per formare un residuo. se la predizione ha

successo l'energia residua è minore rispetto all'originale. In maniera simile la predizione di un campione o una regione dell'immagine può essere effettuata da un campione precedentemente trasmesso nell'immagine stessa. La predizione spaziale è spesso descritta come *Differenzial Pulse Code Manipulation* un termine ripreso da un metodo di codifica differenziale utilizzato nei sistemi di telecomunicazioni. Se il frame è processato in ordine di rasterizzazione allora per ogni pixel dell'immagine si hanno a disposizione il pixel attiguo precedente sulla stessa linea ed i pixel attigui superiormente sulla riga precedente che sono già stati decodificati; sulla base di questi valori si cerca di formare una predizione del pixel corrente un esempio di predizione del pixel può essere il seguente:

$$P(X) = \frac{(2A + B + C)}{4}$$

dove A è il pixel precedente il pixel corrente sulla stessa riga, B il pixel superiore, e C il pixel immediatamente successivo a B ed il residuo trasmesso è

$$R(X) = X - P(X)$$

Se il processo di codifica è con perdita allora i pixel decodificati possono essere (e generalmente lo sono) diversi rispetto a quelli originali, ed il processo di recupero della predizione genera un disadattamento cumulativo (*drift*), un modo di evitare questo problema è utilizzare i pixel decodificati per effettuare la predizione, in questo modo si evita il problema del drift poiché sia il codificatore che il decodificatore calcolano la predizione nello stesso modo. l'efficienza della compressione in questo modo dipende dall'accuratezza della predizione, se è accurata allora l'energia residua sarà bassa, generalmente però non è possibile scegliere un predittore che lavori adeguatamente per tutte le aree di una regione complessa, una migliore predizione può essere ottenuta adattando la predizione a seconda delle statistiche locali dell'immagine, è necessario per il codificatore indicare la scelta del predittore al decodificatore così da permettere una corretta ricostruzione, ciò tuttavia necessità una trasmissione di un maggior numero di bit ed introduce un tradeoff tra l'efficienza della predizione e la qualità della compressione.

2.1.2.2 Transform Coding

Lo scopo del passo di trasformazione è convertire l'immagine o il frame residuo in un diverso dominio dei dati, la scelta della trasformazione dipende da vari fattori:

1. I dati nel dominio trasformato devono essere scorrelati (separati in componenti con la minima inter-dipendenza) e compatti (l'energia deve essere concentrata in un piccolo numero di valori).
2. La trasformazione deve essere reversibile.
3. Deve essere computazionalmente trattabile.

Sono state proposte molte trasformazioni e le più popolari tendono ad essere raggruppate in due grandi famiglie, block-based e image-based; del primo gruppo fanno parte la *Karhunen-Loeve Trasformation* (KLT), *Singular Value Decomposition* (SVD) o *Discrete Cosine Transform* (DCT); esse lavorano su una piccola porzione dell'immagine ed hanno il vantaggio di avere una minore richiesta di memoria oltre a lavorare bene con metodi di predizione basata su blocchi ma introducono degli artefatti ai bordi. Del secondo gruppo il metodo più popolare è la Discrete *Wavelet Trasform* (DWT o semplicemente Wavelet) che opera sull'intera immagine.

2.1.2.3 DCT

La Discrete Cosine Transform opera su X, un blocco di NxN campioni, e crea Y, un blocco di NxN coefficienti; l'azione della DCT può essere descritta come segue:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases}$$

E la sua inversa:

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

L'output della trasformazione bidimensionale è un insieme di NN coefficienti che rappresentano il blocco di dati dell'immagine nel dominio della DCT, e questi coefficienti sono considerati i pesi di un insieme di standard basis patterns. I basis patern sono combinazioni delle funzioni coseno orizzontali e verticali; il valore in alto a destra è la funzione base del coefficiente "DC" e rappresenta la frequenza spa-

ziale zero, avanzando nella riga le funzioni di base incrementano il contenuto relativo alla frequenza spaziale, mentre lungo le colonne aumenta la frequenza spaziale verticale. Ciascun blocco dell'immagine può essere ricostruito combinando gli NxN basis patterns, con ciascuna base moltiplicata per l'appropriato peso (coefficiente).
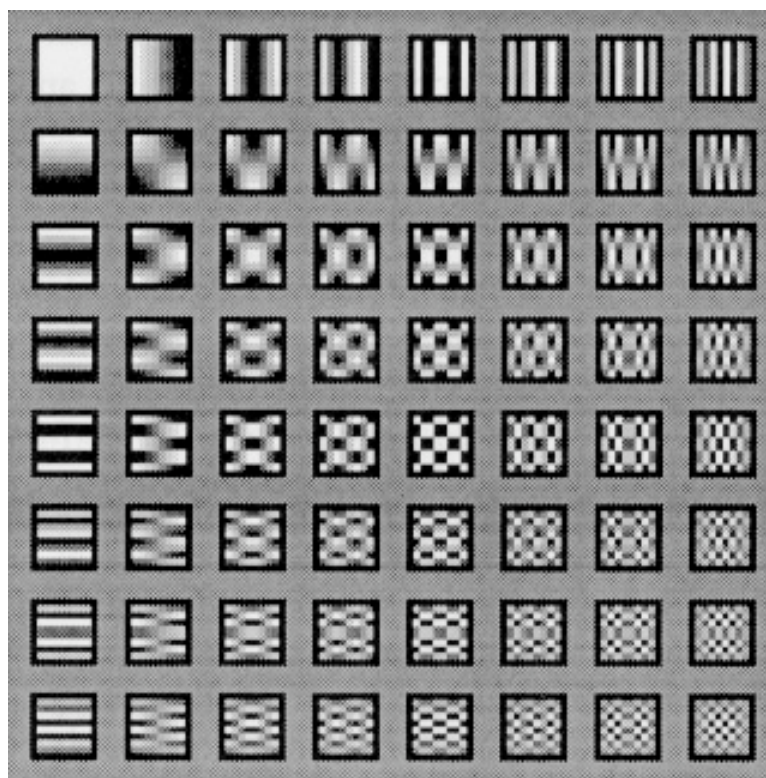
Immagine 2.4 - DCT Basis Pattern

2.1.2.4 Wavelet

La popolare trasformazione wavelet largamente utilizzata nella compressione delle immagini è basata su un insieme di filtri con coefficienti che sono equivalenti alla funzione discreta di wavelet. Le operazioni di base di una trasformazione wavelet sono le seguenti, applicate ad un segnale discreto contenente N campioni. Una coppia di filtri applicati a ciascun segnale per decomporli in una banda a bassa frequenza (L) ed una ad alta (H). Ogni banda è sottocampionata di un fattore due, cioè ciascuna contiene N/2 campioni; con la corretta scelta di filtri quest'operazione è reversibile. Quest'approccio può essere esteso ad un segnale bidimensionale; ogni riga dell'immagine 2d è filtrata con un filtro passa basso ed uno passa alto, e l'output di ciascun filtro è sotto-campionato di un fattore due, per produrre le immagini intermedie L (contenente le basse frequenze dell'immagine originale sotto-campionate nella direzione *x*) ed H (contenente le alte frequenze); il passo successivo si ripete lo stesso procedimento scansionando le immagini ottenute per

colonne, in questo modo si ottengono quattro sotto-immagini (LL, LH, HL, HH), queste quattro immagini di sotto-banda possono essere combinate per creare l'immagine di output con lo stesso numero di campioni dell'originale. 'LL' è l'immagine contenente le basse frequenze in entrambe le direzioni, 'LH' contiene le basse frequenze nella direzione orizzontale e le alte frequenze residue in verticale, 'HL'è l'opposto, contenete le alte frequenze in orizzontale e le basse in verticale, infine 'HH' contiene le alte frequenze in entrambe le direzioni; le immagini di sotto-banda contengono tutte le informazioni contenute nell'originale ma per via della loro natura sparsa sono più facilmente soggette a riduzione. in un'applicazione di compressione dell'immagine la sotto-immagine LL viene sottoposta nuovamente al processo descritto e da essa si ricavano delle nuove sotto-immagini; l'immagine risultante contenete le basse frequenze è filtrata iterativamente un albero di sotto-bande. Molti dei campioni (coefficienti) nella sottobande relative alle alte frequenze sono vicine alo zero (tendenti al nero) ed è possibile ottenere una buona compressione rimuovendo questi coefficienti insignificanti.
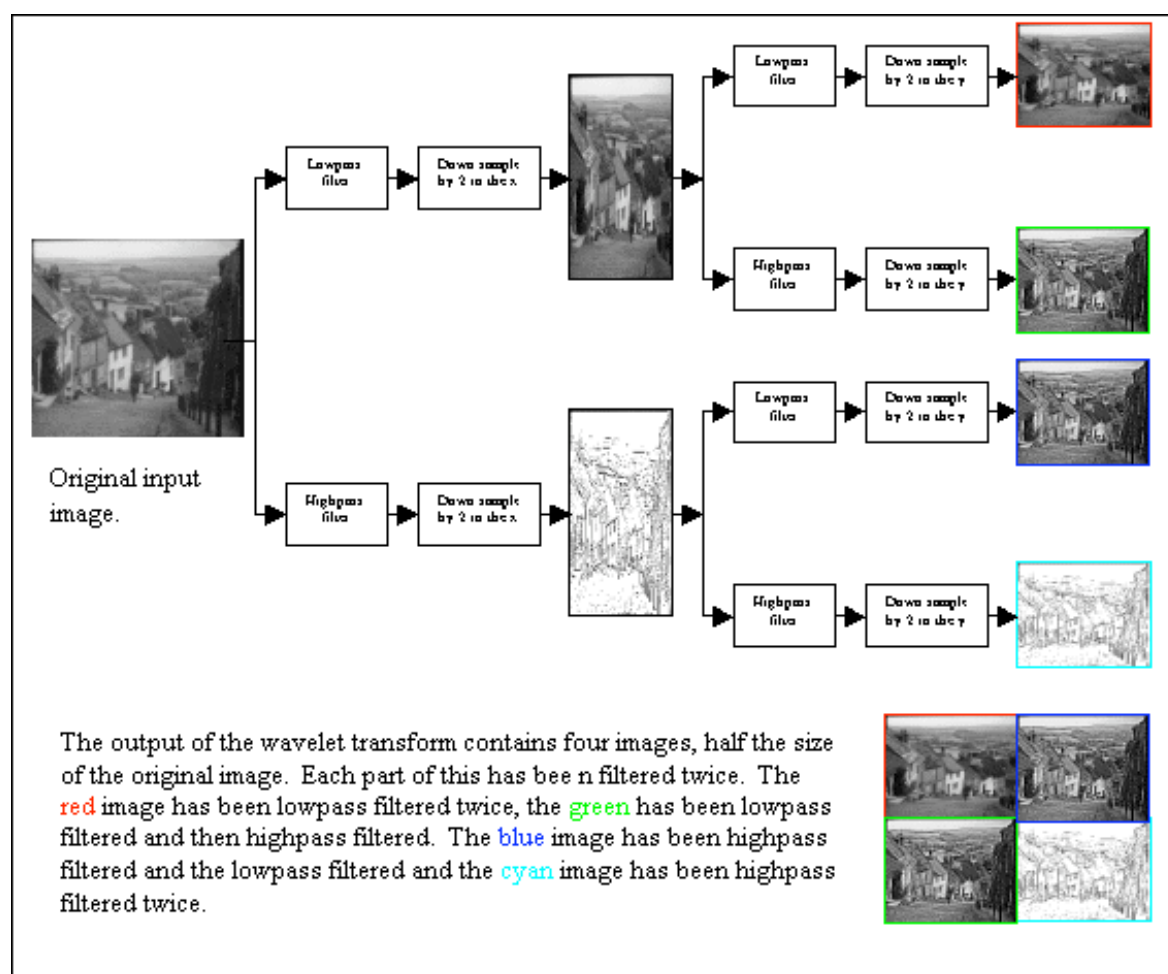
Immagine 2.5 - Wavelet

2.1.3 Quantizzazione

Un quantizzatore mappa il segnale con un range di valori X in un segnale quantizzato con un range di valori Y ridotto; dovrebbe essere possibile rappresentare il segnale quantizzato con un numero di bit ristretto rispetto all'originale. Un quantizzatore scalare mappa un campione del segnale di input in un output quantizzato, un quantizzatore vettoriale mappa un intero gruppo di input in un gruppo di valori quantizzati.

2.1.3.1 Scalar quantizzation

Un semplice esempio di quantizzazione consiste nell'approssimare un numero frazionario all'intero più vicino, il processo è lossy dato che è impossibile sapere con esattezza quale fosse il numero originale partendo dal numero arrotondato; un più generale esempio di quantizzatore uniforme è FQ= *round*(X/QP) Y=FQ·QP. Nella compressione di immagini e video l'operazione è generalmente costruito in due parti, un quantizzatore forward FQ nel codificatore ed un quantizzatore inverso nel decodificatore (dato che la quantizzazione non è reversibile un temine più corretto per il dispositivo è *rescaler*) Un parametro critico è l'ampiezza del passo QP tra due valori scalati successivi, se lo step è largo il range dei valori quantizzati è basso è può essere facilmente rappresentato, in compenso i valori riscalati sono una rozza approssimazione del segnale originale; uno step basso migliora la qualità della ricostruzione ma il largo range dei valori quantizzati riduce l'efficienza della compressione. La quantizzazione può essere utilizzata per ridurre la precisione dell'immagine dopo aver applicato la trasformazione come la DCT o la wavelet rimuovendo valori vicini allo zero in modo da conservare un numero ridotto di coefficienti significativi.

2.1.3.2 Vector Quantizzation

Questo metodo consiste nel mappare un insieme di input (come un blocco di immagini campione) in un singolo valore che, a livello di decodifica, è rimappato in un vettore che approssima il dato originale. L'insieme di vettori è conservato dal codificatore e decodificatore in un codebook. Una tipica applicazione della quantizzazione vettoriale consiste in:

1. Partizione dell'immagine in regioni (es. MxN blocchi)
2. Selezione di un vettore dal codebook con la maggiore corrispondenza con la regione selezionata.

3. Codifica del vettore tramite un intero che indica l'indice della vettore scelto nel codebook.
4. Il decodificatore ricostruisce un approssimazione della regione partendo dal vettore selezionato.

2.1.3.3 Reordering and Zero Encoding

Il risultato della quantizzazione è un vettore sparso contenete alcuni valori non nulli e molti zeri, il riordino (raggruppare assieme coefficienti non nulli) e l'efficiente rappresentazione dei coefficienti nulli sono il passo finale del processo di quantizzazione, utilizzati per semplificare le codifiche successive. *DCT:* i coefficienti significativi della DCT in un blocco sono tipicamente raggruppati attorno alla posizione (0, 0) (altrimenti chiamata coefficiente DC) e rappresentano le 'basse frequenze', la distribuzione è altamente simmetrica nelle direzioni orizzontale e verticale; in un campo residuo i coefficienti sono raccolti intorno alla posizione DC ma sono 'deviati' presentando un maggior numero di valori diversi da zero lungo il lato sinistro dell'area, dovuto a componenti ad alta frequenza lungo l'asse verticale. In base alla distribuzione dei coefficienti viene selezionato il cammino di riordino (l'ordine di scannerizzazione del blocco) ottimo, per una tipica distribuzione un ordine molto utilizzato consiste in un camino a zig-zag, nel caso dei blocchi basati su campi invece si preferisce una scannerizzazione verticale.
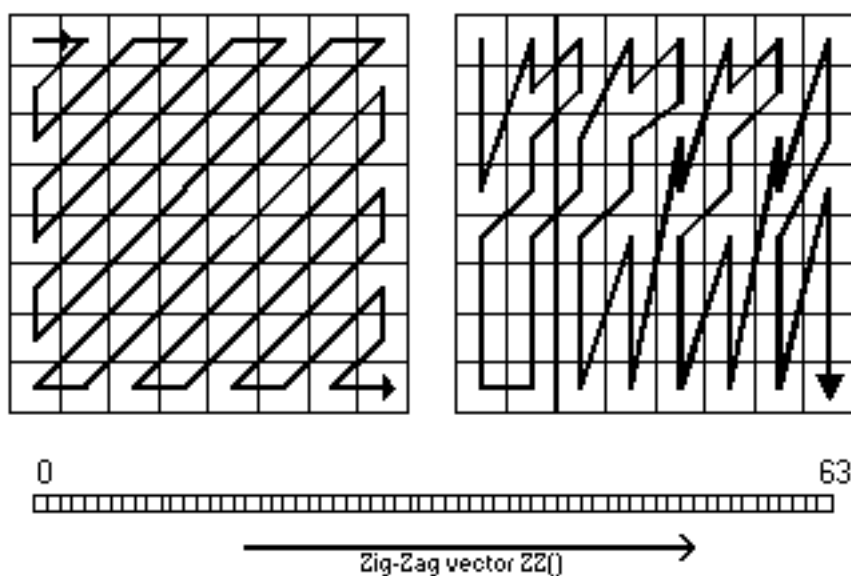
Immagine 2.6 - Zig Zag scan (sinistra) e Vertical Scan (destra)

Il cammino di riordino consiste nello scansire l'intero blocco partendo dal coefficiente DC secondo l'ordine scelto e ricopiarne il contenuto in un vettore monodimensionale in modo da raggruppare i coefficienti non nulli seguiti da lunghe se-

quenze di zeri. Il passo successivo consiste nell'utilizzare una rappresentazione delle sequenze di zeri più compatta, ad esempio rappresentando il vettore come una serie di coppie (*run, livello*) in cui run indica il numero di zeri che precedono un coefficiente non nullo, mentre il livello indica il coefficiente stesso. Dato che le DCT alle alte frequenze sono spesso quantizzate come zeri, e così il vettore riordinato tendenzialmente terminerà con una sequenza nulla, in tal caso è necessario introdurre un valore che indichi la fine del blocco.

12, 6, 6, 0, 4, 0, 0, 0, 3, ...0 ---> (12), (6), (6), (1, 4), (0, 3) EOB

Immagine 2.7 - Esempio di RunLenght Encoding

2.1.3.4 Wavelet

Anche nel caso delle trasformazioni wavelet le sotto-bande ad alte frequenze (vicine al lato in basso a destra) sono vicine allo zero, e possono essere annullate dalla quantizzazione senza una significativa perdita di qualità. I coefficienti non nulli tendono a corrispondere a strutture nelle immagini, perciò un coefficiente è diverso da zero nella sottobanda a bassa frequenza c'è un elevata probabilità che anche il coefficiente corrispondente alla stessa regione nelle sottobande ad alta frequenza sarà non nullo, questo ci permette di costruire un albero di coefficienti indicando come radice il valore nella sotto-banda a frequenza più bassa, a cui si associano i coefficienti corrispondenti alla stessa regione nel layer successivo, ed ad ogni nuovo layer della trasformazione si aggiunge un nuovo livello dell'albero contenente i rispettivi coefficienti rispetto alla regione di partenza (il numero di coefficienti ad ogni nuovo layer aumenta data la maggiore risoluzione).
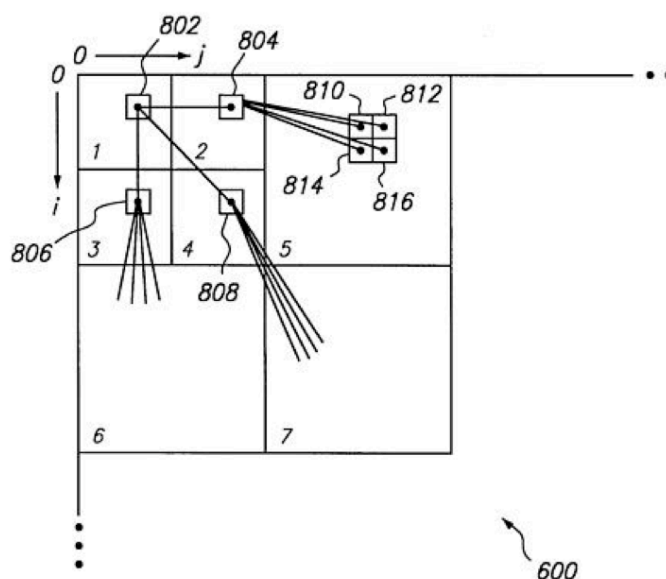
Immagine 2.8 - Quantizzazione Wavelet

Per codificare tali coefficienti nel modo più compatto possibile in modo efficiente si codifica ogni albero di valori, per prima cosa è codificato il coefficiente al layer più basso seguito dai suoi figli al layer successivo e così via. Il processo continua fino a raggiungere uno zero, dato che figli di un coefficiente a zero saranno anch'essi nulli, non è necessario iterare ulteriormente e tutti questi valori sono codificati da un singolo codice che identifica un albero di zeri. Eventualmente può essere prevista l'opzione di continuare l'iterazione anche dopo un coefficiente nullo nel caso sia seguito da coefficienti non nulli; il caso non si verifica molto spesso.

2.1.4 Entropy Coder

Un codificatore entropico converte una serie di simboli che rappresentano gli elementi della sequenza video in un bitstream compresso utilizzabile per le trasmissioni e la conservazione. I simboli di input includono i coefficienti della trasformazione quantizzati, vettori del moto, marker (codici utilizzati per la sincronizzazione), header e informazioni supplementari. Di seguito discuteremo la codifica a lunghezza variabile di Huffman, uno dei metodi più utilizzati per queste applicazioni.

2.1.4.1 Codifica a lunghezza variabile

Un codificatore a lunghezza variabile mappa i simboli in input in una serie di codewords chiamate VLC (*variable lenght codes*); la base di questo tipo di codifica consiste nell'utilizzare codici brevi per la rappresentazione di simboli frequenti, al contrario a simboli meno comuni sono assegnati codici di maggiore lunghezza; in caso di un numero abbastanza ampio di simboli ciò comporta una riduzione della quantità di bit necessari. Tra i vari metodi il più noto ed utilizzato è la *Codifica di Huffman*, che assegna le VLC sulla base delle probabilità dell'occorrenza di ciascun simbolo e nel costruire un insieme di codeword.

$$s_1 = \mathbf{000} \rightarrow p_1 = 0.4$$

$$s_2 = \mathbf{001} \rightarrow p_2 = 0.2$$

$$s_3 = \mathbf{010} \rightarrow p_3 = 0.2$$

$$s_4 = \mathbf{011} \rightarrow p_4 = 0.1$$

$$s_5 = \mathbf{100} \rightarrow p_5 = 0.1$$

Immagine 2.9 - Codifica di Huffman passo 1

La costruzione di un codice di Huffman avviene in tre passi successivi:

1. Il messaggio sorgente deve essere letto una prima volta, in modo da determinare le probabilità associate ad ogni simbolo.
2. Successivamente si costruisce una tabella in cui, nella prima colonna, i simboli del messaggio sorgente sono ordinati in ordine decrescente di probabilità. Si raggruppano quindi i due simboli meno probabili sommando le loro probabilità, e riportando il risultato in una nuova colonna, riordinando se necessario, in modo che lo probabilità siano sempre in ordine decrescente. Il processo viene ripetuto fino a quando non rimangono che due soli valori di probabilità, e viene chiamato processo di riduzione.
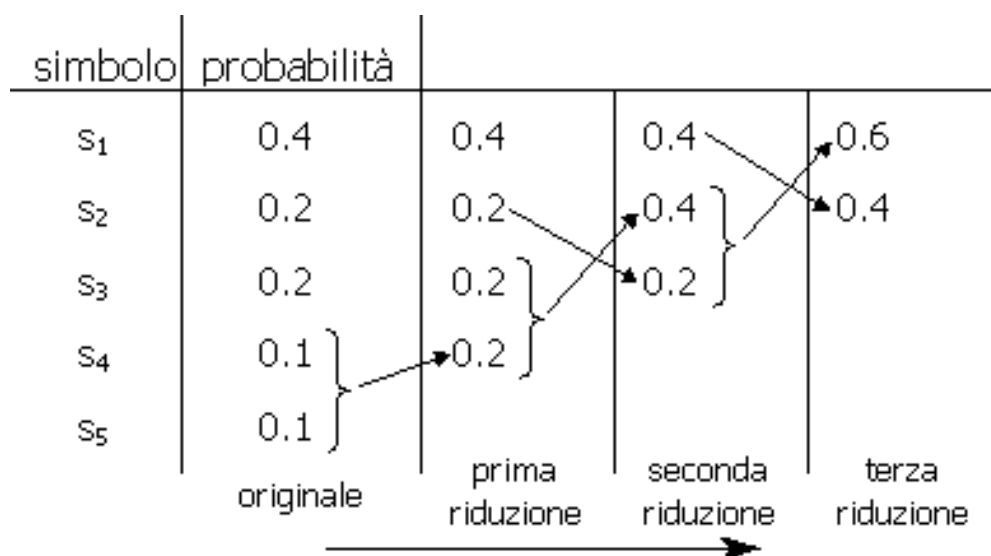
Immagine 2.10 - Codifica di Huffman passo 2

3. Il codice viene infine effettivamente generato nel processo detto di separazione. Nell'ultima colonna dalla tabella generata al punto precedente erano rimasti due soli valori: questi si possono codificare semplicemente attribuendogli i bit 0 e 1.
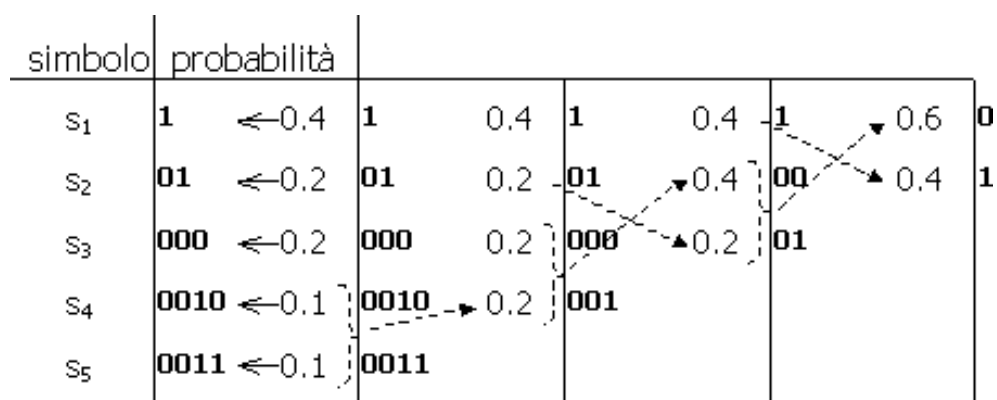
Immagine 2.11 - Codifica di Huffman passo 3

Passando alla colonna immediatamente a sinistra, uno dei valori di probabilità presenti sono nuovamente separati in due e la codifica avviene aggiungendo il bit 0 al primo e il bit 1 al secondo.

2.2 Codec Video

Un codec video codifica un immagine sorgente o una sequenza video in una forma compressa e la decodifica per produrre una copia o approssimazione della sequenza sorgente. Il codec rappresenta la sequenza video originale tramite un modello (un efficiente rappresentazione codificata che può essere utilizzata per la ricostruzione approssimata dei dati video) idealmente il modelle dovrebbe rappresentare la sequenza utilizzando il minor numero di bit e la maggior fedeltà possibili; questi due obiettivi sono generalmente in conflitto. un video encoder consiste in tre unità funzionali principali, un modello temporale, un modello spaziale e un codificatore entropico, l'input del modello temporale è una sequenza video non compressa, esso cerca di ridurre la ridondanza temporale sfruttando la similarità tra due video frame vicini tra loro, generalmente costruendo una predizione del frame corrente; l'output del modello è un frame residuo creato sottraendo la predizione dal frame vero e proprio ed un insieme di parametri del modello, tipicamente un insieme di vettori di moto che indicano come il moto sia compensato. I frame residui formano l'ingresso al modello spaziale che fa uso delle similarità tra i campioni vicini all'interno di un frame per ridurne la ridondanza spaziale, nello standard MPEG ciò è ottenuto applicando una trasformazione ai campioni residui e quantizzando i risultati, la trasformazione converte i campioni in un altro dominio, in cui sono rappresentati dai coefficienti della trasformazione, che vengono poi quantizzati per rimuovere valori insignificanti, lasciando un minori numero di coefficienti significativi per ottenere una più compatta rappresentazione del frame residuo, l'output del modello spaziale è l'insieme dei coefficienti quantizzati della trasformazione. Infine i parametri del modello temporale (i vettori del moto) e di quello spaziale (i coefficienti) vengono poi compressi da un codificatore entropico il cui compito è rimuovere le ridondanze statistiche nei dati (ad esempio rappresentando i valori con maggiore occorrenza con codici binari di lunghezza minore) e producendo uno stream di bit o un file che possono essere trasmesse e/o memorizzate. Una sequenza compressa consiste in un header e nell'insieme dei parametri spaziali e temporali codificati. Il video decoder ricostruisce il video a partire dal bitstream compresso, i coefficienti ed i vettori del moto sono decompressi prima da

un decompressore entropico, che inverte le compressioni statistiche, in seguito da un decompressore spaziale che ricostruisce i frame residui ed infine dal decompressore temporale che utilizza i vettori del moto per ricostruire la sequenza delle immagini. Un codec è un programma o un dispositivo che si occupa di codificare e/o decodificare digitalmente un segnale (tipicamente audio o video) perché possa essere salvato su un supporto di memorizzazione o richiamato per la sua lettura. Oltre alla digitalizzazione del segnale, i codec effettuano anche una compressione (e/o decompressione in lettura) dei dati ad esso relativi, in modo da poter ridurre lo spazio di memorizzazione occupato a vantaggio della portabilità o della trasmissibilità del flusso codificato. I codec si dividono in base alla modalità in cui effettuano la compressione: *con perdita di informazione (lossy)* o *senza perdita di informazione (lossless)* Per realizzare tale compressione si fa ricorso alla riduzione della precisione dei colori dei singoli pixel (codec video) o delle frequenze da riprodurre (in alcuni codec audio vengono soppresse le frequenze non udibili dall'orecchio umano), alla eliminazione delle ridondanze o alla scrittura delle sole differenze (codec video) rispetto ad una immagine di riferimento. Esistono vari tipi di codec, differenti tra loro per il tipo di segnale su cui devono operare e per l'algoritmo di codifica/compressione in essi implementato.

2.3 Formato MPEG

il Moving Picture Expert Group, un comitato di standard internazionali ha prodotto una serie di standard per la compressione audio e video che sono ottimi esempi di strategie sofisticate e anche commercialmente importanti. alcune delle più importanti applicazioni dello standard MPEG includono la memorizzazione su disco, la trasmissione broadcast di video digitali, ed altro. L’algoritmo video MPEG è un una versione altamente rifinita di una popolare ed efficace classe di algoritmi di compressione video Motion-Compensated Discrete Cosine Transform (MC-DCT) l’algoritmo utilizza:

- *predizione temporale:* tecniche di predizione e compensazione del moto per eliminare le ridondanze tra le varie immagini di una sequenza.
- *decomposizione nel dominio della frequenza:* l’uso della DCT per decomporre i blocchi spaziali dell’immagine in modo da eliminare le ridondanze statistiche e percettive.
- *quantizzazione:* selettiva riduzione nella precisione con cui l’informazione è trasmessa per ridurre il bitrate minimizzando la perdita di qualità.

- *codifica a lunghezza variabile:* per eliminare le ridondanze statistiche nelle sequenze simboliche risultanti dalla quantizzazione.

il video che elabora l'MPEG è composto di una sequenza di una sequenza di frame o field di luma e chroma: l'MPEG-2 è ottimizzato per un'ampia classe di rappresentazioni video che includono le sequenze basate su campi (interlacciate). Per le sequenze basate su campi l'MPEG-2 si aspetta che il chroma associato con ciascun campo sia sottocampionato verticalmente all'interno del campo stesso. Lo standard MPEG è inteso per essere generico, cioè in grado di supportare le necessità di molti tipi di applicazioni, può essere considerato come un toolkit per la compressione video da cui ciascun utente sceglie le caratteristiche necessarie per lo specifico uso; lo standard MPEG è principalmente una specifica di un bitstream, inoltre contiene la specifica di un tipico processo di decodifica per aiutare l'interpretazione del bitstream, lasciando ai vari sviluppatori la costruzione di specifici strumenti per eseguire la compressione.
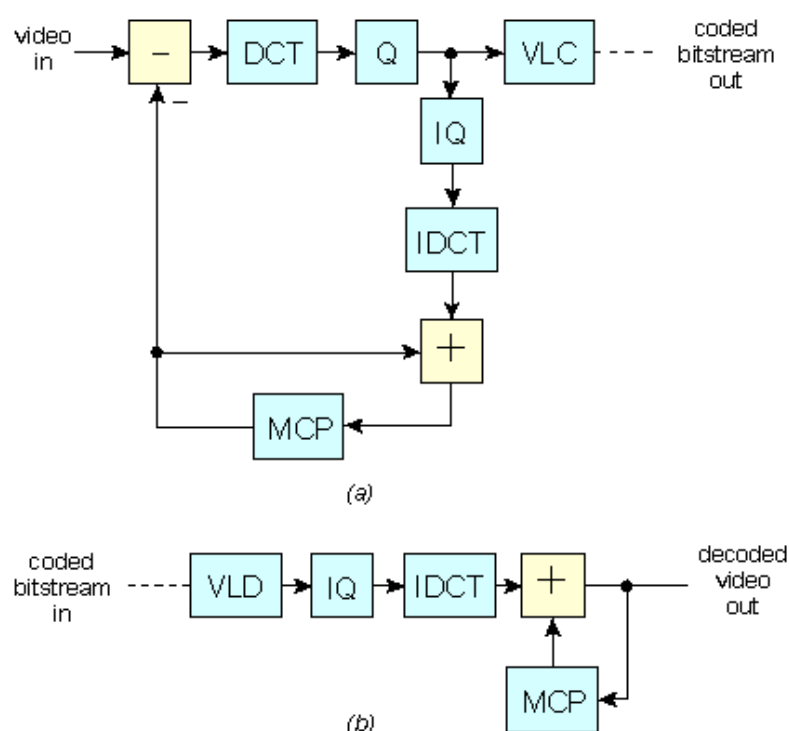
Immagine 2.12 - MPEG processo di codifica(a) e decodifica(b)

2.3.1 MPEG-2

MPEG-2 è uno standard introdotto nel 1994 da MPEG; è un sistema di codifica digitale definito per sorgenti audio,video, e specifica il formato di multiplexing e trasporto per servizi multimediali diffusivi a qualità televisiva o superiore. L'MPEG-2 è stato destinato al broadcast televisivo, fin dalla sua introduzione nel 1994,

un'efficiente codifica per il video interlacciato e la scalabilità sono state le caratteristiche che hanno permesso di digitalizzare efficacemente i segnali televisivi, grazie ad esso si ottengono immagini televisive di buona qualità con bitrate compresi tra 4 e 9 Mbit/s.

Lo standard MPEG-2 è specificato nell'ISO/IEC 13818 ed è suddiviso in diversi documenti, i principali sono i seguenti:

- Part 1 - Systems: descrive la sincronizzazione e il multiplexing tra audio e video.
- Part 2 - Video: codec per segali video interlacciati o progressivi.
- Part 3 - Audio: codec per la compressione percettiva di segnali audio, ed un estensione dell'MPEG-1 che permette il multichannel.

In questo documento ci concentreremo principalmente sui primi due.

2.3.1.1 MPEG Video

Il modello di predizione temporale utilizzato dall'MPEG si basa sulla stima del moto basata su macroblocchi, un macroblocco corrisponde ad ad una regione di 16x16 pixel, ed è l'unità di base per la predizione della compensazione del moto, un molti standard, inclusi l'MPEG-2 e 4 su cui focalizziamo la nostra attenzione; per un materiale video nel formato 4:2:0 un macroblocco è organizzato da una regione di 16x16 pixel rappresentato da 256 campioni di luminanza, disposti in quattro blocchi di 8x8 pixel, e da due gruppi di 64 (8x8) campioni di crominanza, uno per la rossa ed uno per la blu, per un totale di sei blocchi.
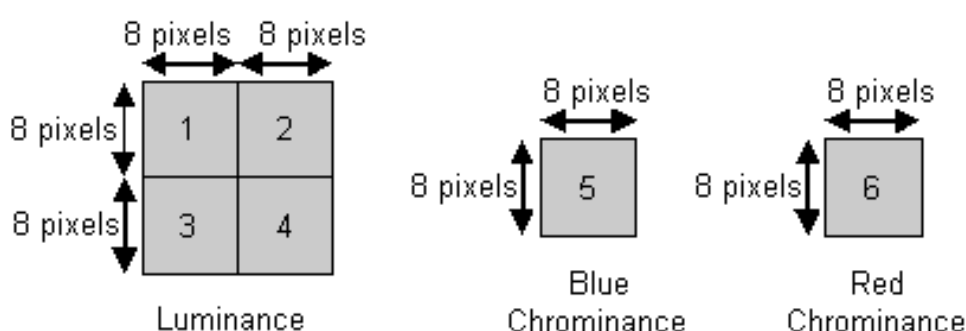
Immagine 2.13 - Macroblocchi MPEG-2

Una regione di queste specifiche dimensioni permette un ragionevole compromesso tra il calcolo di approssimazione di arbitrari motion field (che divengono più accurati al diminuire della dimensione dei blocchi), e l'overhead generato dalla necessità di trasmettere i vettori di moto. La stima del moto implica trovare una regione di 16x16 nel frame di riferimento che combacia maggiormente con il macroblocco corrente; mentre la compensazione del moto, partendo dalla la regione se-

lezionata nel frame di riferimento, produce il macroblocco residuo (sia per i valori di luma che di chroma) sottraendola dalla regione corrente, che è codificato e trasmesso assieme al vettore del moto che descrive offset tra la posizione del blocco migliore ed il blocco corrente. Le meccaniche della compensazione del moto utilizzano il vettore per estrarre i blocchi predetti dalla pittura a cui si fa riferimento, sottrarli, a passare la differenza ad un'ulteriore compressione, Il vettore scelto per il macroblocco è applicato direttamente all'immagine per determinare la predizione della luma, ma è scalato della metà in entrambe le direzioni (corrispondenti al tipo di sotto campionamento) prima di essere applicati al predittore di crominanza. Il processo di stima del moto è generalmente l'attività più costosa in un codificatore MPEG, ma ha un sostanziale impatto sull'efficienza della compressione. L'approccio straighforward alla stima del moto è di valutare ciascun vettore del moto e selezionare il migliore, non è comunque efficace nei campi di moto complicati dall'interessamento di sostanziali rotazioni, ingrandimenti o deformazioni; in questi casi, e nei tagli di scena, si tiene conto della bassa sensibilità dell'occhio umano alla distorsione in materiali tanto complicati, così come alla compressione prodotta da altri elementi dello standard. Il criterio applicato per la scelta del macroblocco migliore implica la misura e la minimizzazione l'errore assoluto medio (MAE) tra il macroblocco corrente ed il macroblocco a cui è specificato l'offset (spesso valutato sul solo componente di luma) a la ricerca è strutturata in maniera da ridurre il numero totale di paragoni necessari. La strategia più comune è una ricerca a base piramidale, che conta sul formare una collezione di immagini costruite con un filtro passa-basso incrementale e sottocampionate per la picture corrente e quella di riferimento; la ricerca comincia comparando le regioni tra la versione a risoluzione minore delle immagini, il risultato inizializza una ricerca dello stesso tipo ad una risoluzione più alta secondo una serie di passi; tale ricerca può richiedere molte comparazioni minori finché la ricerca localo può essere effettivamente ristretta, sono efficaci in molti generi ma può non esserlo se la differenza tra una buona e cattiva corrispondenza della stima del moto è ristretta ai dettagli che non appaiono nell'immagine a risoluzione minore. L'efficienza della compensazione può essere aumentata permettendo la ricerca di un'efficace regione di predizione nella picture di riferimento includendo non solo le posizioni intere dei pixel ma anche quelle frazionarie, (*integral & fractional pixel offset*) per un pixel offset frazionario la predizione dei macroblocchi è costruita tramite un interpolazione lineare dei valori re-

lativi ai pixel adiacenti (p=a+b+c+d/4). L'MPEG permette l'interpolazione dei mezzi pixel sia in verticale che in orizzontale, o in entrambi, la rozzezza dell'interpolazione non comporta una traccia del moto perfetta ad alte risoluzioni ma porta benefici di un efficace filtro passa basso nel contenuto dell'immagine nei punti d'interpolazione, questo a volte rimuove sostanziali rumori da un immagine riferita a provvede un migliore predittore di quello che ci si aspetterebbe.

2.3.1.2 Tipi di Immagini e Macroblocchi di predizione

Una delle chiavi del MPEG è il ragionevole supporto all'accesso casuale, l'accesso casuale in una sequenza arbitraria moto-compensata è difficile, il ricevitore necessità di molta informazione sui cambiamenti da un immagine all'altra eventualmente fino all'inizio della catena, il che può significare un'enorme quantità di calcolo, inoltre un eventuale errore anche minimo in uno dei frame tenderà a propagarsi fino alla fine della catena; per risolvere tali problemi periodicamente si inserisce nella sequenza un immagine priva di predizione temporale, in modo che tutte le immagini successive facciano riferimento ad essa piuttosto che all'immagine iniziale; considerata la verosimiglianza ottenuta dal regolare uso delle immagini non predette l'MPEG fa uso di nuove strategie di predizione che migliorano ulteriormente l'efficienza della compressione e sviluppano il concetto delle predizione forward e backward. Per prima cosa si definisce un'immagine ancora, come una utilizzata come base per la predizione, in seguito si definiscono quattro diverse strategie di predizione che possono essere utilizzate in uno specifico macroblocco. Ogni immagini appartiene ad un tipo dichiarato che limita il tipo di macroblocco permesso nella stessa.

Tipo di macroblocco	predizione
non predetto	nessuna
backward predicterd	si riferisce all'immagine ancora seguente temporalmente più vicina
forward predicted	si riferisce all'immagine ancora precedente temporalmente più vicina
bidirectionally predicted	media della predizione tra le immagini ancora più vicine temporalmente sia precedenti che seguenti

tipo di Immagine	Ancor Pic-ture	Tipo di Macroblocco Accessibili
I-Picture	Si	non predetto
P-Picture	Si	non predetto, forward predicted
B-Picture	No	tutti

L'MPEG distingue quattro tipi di immagini, per il procedimento, questo per le richiesta conflittuale di un buon rapporto di compressione e della necessità dell'accesso casuale alla sequenza:

- **I-Frames (Intracoded frames):** sono auto contenuti, cioè codificate senza alcun riferimento ad altre immagini della sequenza, un frame-I è compresso come un'immagine fissa, e definisce un punto per l'accesso casuale. I frame-I utilizzano blocchi di 8x8 pixel su cui si applica la trasformazione DCT sui cui coefficienti viene poi eseguita una predizione spaziale (DPMC) e le differenze tra due successivi blocchi di un componente sono trasformate utilizzando codifiche a lunghezza variabile. In questi elementi non viene sfruttata la ridondanza temporale ma esclusivamente quella spaziale, l'utilizzo di frame-I periodici facilita l'inizializzazione per il ricevente, il decoder inoltre può avvantaggiarsi dalla presenza di un intraframe quando si verifica un errore sul canale impossibile da correggere, frame-I periodici sono importanti nei flussi multimediali in quanto consistono in punti di accesso per il fast-forward ed i comando di salto; la frequenza di frame-I è variabile e decisa dal codificatore e permette di adattare l'efficienza della compressione con la necessità dell'accesso casuale o delle modalità di scorrimento veloce del video.
- ***P-Frames (predicted-coded frames):*** i frame-P sono predetti temporalmente solo in una direzione (forward) e richiedono per essere decodificati l'informazione del precedente frame-I o P; un frame-P può includere porzioni che sono esclusivamente intra-codificate.
- ***B-Frames (Bi-directionally predictive-coded frames):*** il frame-B è un tipo di immagine che include una predizione dia da un frame precedente che da un frame futuro, chiamato anche anchor-frames e sono frame di tipo I o P; la base della predizione del frame-B consiste nella correlazione del con immagini avvenute nel passato, così come in quelle che avverranno nel futuro. I frame-B incrementano l'efficienza della codifica.

- ***D-Frames (DC-coded frames):*** l'ultimo tipo di frame è utilizzato per le modalità di fast-forward o fast-rewind, consiste in frame intra-codificati di cui si memorizzano solo le basse frequenze, non sono ampiamente utilizzati dato che possono essere sostituiti dai frame-I.

Sebbene non richiesto dallo standard un insieme di sequenza in cui vi è uno spazio fissato tra i frame-I ed un altro tra le immagini ancora (frame I e P) è largamente utilizzato, ed in genere ad esso sono assegnati due parametri, il primo indica il numero di immagini tra due frame-P consecutivi, il secondo tiene traccia del numero di immagini tra un ancora e la successiva. Ci sono numerose implicazioni di questo set di possibili predizioni:

Usare blocchi predetti bidirezionalmente permette un efficace predizione di background scoperti, aree del frame corrente non visibili nel passato ma presenti nel futuro.

- La predizione bidirezionale può provvedere all'interpolazione ad un gradi più raffinato rispetto all'interpolazione dei mezzi pixel per la compensazione del moto, Immaginate un oggetto presente nell'offset 0 in una picture precedente l'ancora e nell'offset ½ in un'immagine ancora adiacente e che segue; usando la predizione bidirezionale si può ottenere l'equivalente di ¼ dell'interpolazione dei pixel dell'oggetto.
- La predizione bidirezionale può ridurre il rumore del predittore, se una buona predizione è possibile sia nell'immagine seguente che precedente allora una media tra le due produce una riduzione del rumore e un incremento dell'efficienza della predizione.
- La predizione bidirezionale incrementa la complessità della stima del moto in due modi: ovviamente richiede il doppio del lavoro per compiere una stima del moto in quando deve compiere il paragone con due immagini invece che con una sola; più significativo comunque è il fatto che poiché le immagini ancora sono separate da un certo numero di altri frame necessitiamo di un overall range più ampio per tenere traccia degli oggetti, ad esempio se un oggetto si muove di 10 pixel per frame è necessaria una stima del moto di 10 pixel per ottenere una buona predizione dell'oggetto in assenza di frame B, con i frame-B presenti invece l'ampiezza necessaria per tenere traccia dell'oggetto incrementa di un fattore M (10xM nel nostro esempio) pari al numero di B-frame presenti tra due ancore consecutive.

- Poiché i B-frame non possono essere utilizzati per predire altre immagini la qualità della compressione è determinata dall'accettazione visiva del risultato.

Per i frame-I e P, poiché la ricostruzione è utilizzata anche per le predizioni, in questo caso la qualità deve tenere conto, oltre alla fedeltà visiva, della fedeltà rispetto alla predizione. un rapporto di 5:3:1 per i frame I, P e B è piuttosto comune; questi rapporti possono variare dinamicamente a seconda delle scene, in una scena statica la le immagini predette saranno di dimensioni molto inferiori rispetto a quelle non predette, grazie alla scarsa differenza tra le immagini, in caso di immagini molto dinamiche invece vi sarà una maggiore dispersione dei bit utilizzati tra i tre tipi di frame. inoltre dato che i frame-B non sono usati nelle predizioni un eventuale errore in esse non andrà a disturbare il resto della sequenza.
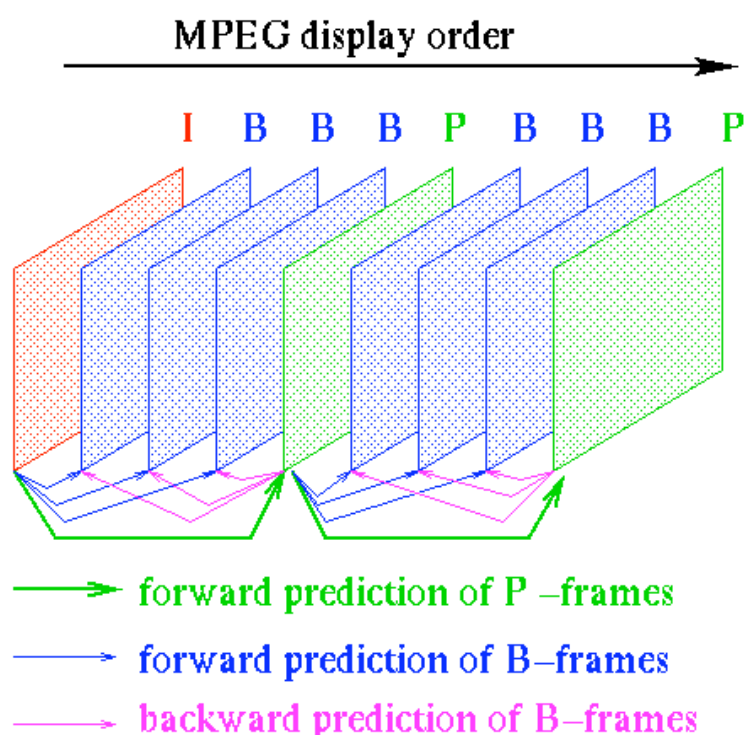
Immagine 2.14 - Group of Frames (GOP) MPEG-2

2.3.1.3 Ordine di trasmissione e Visualizzazione

Combinare l'uso dei frame-B con il desiderio di economizzare l'uso della memoria comporta una differenza nell'ordine in cui i frame verranno trasmessi rispetto a quello in cui saranno visualizzati. Il riordino è inteso a assicurare che, nonostante lo spazio tra le ancore, un decoder necessita solo di avere abbastanza memoria per conservare due immagini ancora più lo spazio per l'immagine che si vuole correntemente visualizzare (*scratch*). Ciò è ottenuto trasmettendo la versione compressa dell'ancora prima di tutti i frame-b che ad essa fanno riferimento.

2.3.1.4 Predizione di Frame e Campi

L'MPEG-2 supporta la modalità interlacciata per la trasmissione e la rappresentazione dei video e permette specificatamente una compressione più efficiente per il materiale interlacciato. un modo in cui ciò è ottenuto è permettendo, per ogni macroblocco, sia la predizione basata su frame che quella basata su campi (*adaptive frame/field compression*), nel caso di predizione basata su campi il macroblocco è diviso nei due sottoblocchi relativi ai campi. Il campo superiore è predetto dai campi superiori delle immagini ancora a cui si riferisce usando i modelli appropriati per il tipo di immagine, così come il campo inferiore fa riferimento esclusivamente ad altri campi inferiori. La predizione basata su campi può essere estremamente efficace in istanze di sostanziale accelerazione orizzontale, in questi casi un oggetto catturato in due differenti campi appare avere un bordo verticale dentellato quando visualizzato come frame, con la profondità della dentellatura variabile a seconda dell'accelerazione. La stima basata su frame è inefficace in questi casi mentre una predizione sui campi può funzionare piuttosto bene. D'altro canto negli altri casi una predizione sui frame può essere più efficace per risolvere la ridondanza verticale, permettendo una scelta del modo di processare un macroblocco di base si ottiene il meglio dei due casi al costo di un lieve incremento di overhead (i vantaggi sono comunque superiori agli svantaggi in termini di efficienza). Questo tipo di predizione è piuttosto importante in alcuni dei materiali più difficili da comprimere, ad esempio nei casi in cui un rapido e complesso movimento in primo piano è accompagnato uno sfondo molto dettagliato con un moto più lento. L'abilità del metodo adattivo di migliorare alcuni di questi casi limite, lo rende molto più valente rispetto a quanto l'indice di miglioramento medio da solo mosti. Una speciale versione della stima del moto basata su campi (*dual prime*) è anche disponibile, essa può essere utilizzata solo per i frame-P in assenza di frame-B tra l'immagine corrente e quella di riferimento. ciascun campo nel macroblocco corrente è predetto utilizzando la media del campo del macroblocco di ciascun campo del frame precedente, piuttosto che inviare quattro vettori separati per identificare l'offset tra ciascuno dei campi di macroblocchi correnti ed i loro predittori nei campi precedenti è inviato un singolo vettore , ed estrapolato per calcolare i quattro vettori utilizzando un modello a velocità costante, sia v_{ij} il vettore del campo i nell'immagine corrente verso il campo j del riferimento, e sia v il vettore trasmesso: $v_{11}=v_{22}=v$, $v_{21}=(3/2)v$, $v_{12}=(1/2)v$. Queste formule possono essere comprese dal

modello a velocità costante, poiché se una regione è tracciata con velocità costante il dispalcement tra i campi 1 e 1 nell'immagine corrente e in quella di riferimento, così come il displacement tra i campi 2 e 2 risulterà identico; il displacement tra il campo 2 rispetto a quello 1 sarà più largo di un fattore 1,5 (poiché la distanza temporale è 1,5 volte maggiore rispetto a quella tra i campi 1 e 1 o 2 e 2); per lo stesso motivo il displacement tra il campo 1 ed il campo 2 sarà ridotto della metà, un offset di ±1 pixel è utilizzato per compiere delle correzioni minori. Il metodo dual prime raggiunge alcuni dei vantaggi dei frame-B attraverso la media tra i campi che può ridurre il rumore e sintetizzare un più fine grado di compensazione del moto.

2.3.1.5 Decomposizione nel dominio della frequenza

Dopo la predizione temporale il blocco residuo, o quello originale nel caso di frame non predetti, è trasformato utilizzando la DCT, per preparare il macroblocco alla trasformazione sei blocchi di 8x8 pixel sono estratti dal macroblocco, quattro dai valori della luma e uno da ciascuno dei valori di chroma; l'estrazione dei blocchi di luminosità può avvenire in due modi diversi, il *frame DCT mode* in cui il macroblocco viene suddiviso in quattro senza modifiche, ed il *field DCT model* in cui le righe (o mezze righe dato che ogni riga viene suddivisa tra due blocchi) vengono inserite in maniera intervallata, in uno le righe pari, nell'altro quelle dispari, in tale modo si ottengono blocchi composti dalle linee destinate a ciascun campo nella modalità interlacciata. lo standard MPEG non specifica esattamente come costruire la trasformazione DCT o la sua inversa, in modo da lasciare la maggior libertà possibile di implementazione dei vari sistemi, piuttosto specifica un test che qualunque implementazione di IDCT deve superare, in cui la IDCT proposta trasforma una serie di blocchi dati e ne confronta il risultato con l'insieme di risultati ideale, se il numero di errori è minore di una certa soglia la trasformazione è accettata.

2.3.1.6 Quantizzazione

Il sistema visivo umano non è uniformemente sensibile all'errore di quantizzazione dei coefficienti; gli errori nei coefficienti DC risultano in distorsioni a valori medi per i corrispondenti blocchi di pixel, che possono esporre i confini del blocco; questo è più visibile di errori in coefficienti relativi ad alte frequenze che appaiono come rumori o texture. Durante la quantizzazione i differenti livelli di importanza percettiva possono essere espressi nell'allocazione dei bit per portare il rumore in aree meno importanti ciò può essere ottenuto variando il passo per i differenti coefficienti, le

basse frequenze saranno calcolate con maggiore finezza, mentre le alte frequenze meno importanti quantizzate più rozzamente. Per questo motivo l'MPEG applica un passo di quantizzazione di dimensione non uniforme per ogni coefficiente, che può variare sia da coefficiente a coefficiente che da macroblocco a macroblocco, infatti il passo è determinato con la seguente equazione: ss=qf[m,n]*qs, il fattore qf[m,n] dipende dalla locazione del coefficiente all'interno del blocco, il fattore qs è il passo di base; questo permette di enfatizzare tra le basse frequenze quelle maggiormente percepite. L'MPEG provvede due matrici dei pesi qf[m,n] di base, per l'uso nei blocchi predetti o non predetti:

*	16	19	22	26	27	29	34	16	16	16	16	16	16	16	16
16	16	22	24	27	32	34	37	16	16	16	16	16	16	16	16
19	22	26	27	29	34	34	38	16	16	16	16	16	16	16	16
22	22	26	27	29	34	37	40	16	16	16	16	16	16	16	16
22	26	27	29	32	35	40	48	16	16	16	16	16	16	16	16
26	27	29	32	35	40	48	58	16	16	16	16	16	16	16	16
26	27	29	34	38	46	56	59	16	16	16	16	16	16	16	16
27	29	35	38	46	56	69	83	16	16	16	16	16	16	16	16

Immagine 2.15 - matrici di Quantizzazione

da notare che per i blocchi non predetti il valore del coefficiente il alto a sinistra non è specificato, questo trattamento inusuale riflette il fatto che l'occhio umano è molto sensibile all'errore nel livello DC dei blocchi non predetti, e tende a riconoscerli velocemente come distorsioni nell'immagine ricostruita, la matrice di default per i blocchi non predetti mostra un forte bias verso verso le basse frequenze, al contrario la matrice dei blocchi predetti è piatta; questo perché la predizione del moto tende ad eliminare la maggior parte delle basse frequenze, comportando benefici limitati nell'enfatizzarle ulteriormente in questa fase. L'MPEG permette al decoder di sovrascrivere queste matrici di default.

2.3.1.7 Codifica a lunghezza variabile

La quantizzazione crea una rappresentazione discreta efficiente dei dati, l'assegnamento di codeword prende i valori così ottenuti e produce dei flussi digitali; ipoteticamente ogni valore può essere facilmente rappresentato da 'parole' di lunghezza fissa o uniforme, ma si può ottenere una maggiore efficienza con l'uso di codificatori entropici il cui compito è di beneficiare delle proprietà statistiche del segnale. Sia i coefficienti quantizzati sia diversi tipi di informazioni di supporto come il tipo dei macroblocchi di predizione o i vettori del moto esibiscono delle concentrazioni statistiche che possono condurre ad un ulteriore diminuzione del bitra-

te utilizzando codifiche a lunghezza variabile. Il tipo di codifica utilizzata dall'MPEG è una versione modificata della codifica di Huffman, dato che essa nella sua forma base ha due svantaggi per un CODEC video pratico, per prima cosa è necessario assicurato che il decoder abbia lo stesso insieme di codewords dell'encoder, il che significa trasmettere la tabella di probabilità o l'albero di codifica, e per alfabeti lunghi ciò introduce un notevole overhead; secondo la tabella di probabilità non può essere calcolata prima della codifica dell'intera sequenza il che comporta un enorme ritardo nell'intero processo. Perciò la dimensione delle tabella di Huffman è generalmente ristretta ad un sottoinsieme che comprende i simboli più probabili, con l'aggiunta di un codice particolare chiamato "ESCAPE", quando al codificatore arriva un simbolo non compreso nella tabella esso viene inserito nel bitstream direttamente preceduto dal simbolo di ESCAPE in modo da riconoscerlo, questo metodo è efficace se la probabilità di trovare una corrispondenza tra i simboli in ingresso e la tabella è abbastanza alta. Un'altra modifica alla codifica è l'eliminazione del codice composto da tutti 0, in modo da evitare problemi dato che lo start code inizia con 23 zeri ed è utilizzato per la resincronizzazione in seguito ad un errore. I coefficienti quantizzati della DCT formano la gran parte del materiale che necessita di essere codificato, empiricamente comunque il blocco di coefficienti quantizzati tende a contenere un elevato numero di zeri, particolarmente in prossimità delle alte frequenze; l'MPEG aumenta questo fenomeno trasformando la matrice in un vettore monodimensionale, ed in seguito applicando la codifica, che può beneficiare del gran numero di zeri, per ottenere un singolo vettore il blocco viene scannerizato in un ordine preciso, l'MPEG-2 dispone di due strategie diverse, il zig-zag scanning o il vertical scanning, quest'ultimo spesso più efficiente specie nel caso sia stata utilizzata la trasformazione basata su campi. una volta che il vettore da 64 elementi è stato creato è utilizzato la codifica *runlenght amplitude*. Per prima cosa il coefficiente DC quantizzato è trattato specificatamente, ricevendo il suo codice di Huffman dedicato, poiché esso tende ad avere un unica caratteristica statistica relativa agli altri coefficienti, inoltre c'è una certa ridondanza tra coefficienti DC adiacenti in un frame non predetto, e solo la differenza tra essi è codificata. I restanti coefficienti sono raggruppati in una serie di coppie, ciascuna delle quali contiene il numero degli zeri in sequenza ed il primo valore diverso da zero incontrato nella scansione del vettore in più viene aggiunto un ulteriore coppia alla fine che indica la fine del blocco (EOB) ad esempio la sequenza 12, 0, 0,

6, 6, 0, 4, 0, 0 è raggruppata in (0,12), (2,6), (0,6), (1,4), (2,EOB); a ciascuno di questi valori è poi assegnato uno specifico codice di Huffman. Ci sono una varietà di informazione di supporto, utilizzate per ridurre il numero di blocchi che devono subire la codifica runeleght-amplitude e per ridurre l'informazione.

2.3.1.8 Syntactical Layering in MPEG

Il bitstream è strutturato in sei livelli gerarchici, in modo da supportare tutte le caratteristiche del video MPEG:

1. ***Sequenza:*** definisce il tipo di frame, la dimensione il rapporto la dimensione dei buffer e vari altri parametri statici, permette inoltre di definire matrici di quantizzazione non statiche.
2. ***Group of Picture:*** un insieme di immagini contigue in ordine di trasmissione che aiuta l'accesso casuale, il primo frame trasmesso deve essere di tipi I, e l'header del GOP può indicare se è aperto o chiuso e se è stato separato dal gruppo precedente, inoltre contiene i valori di temporizzazione per l'inizio del GOP; questa struttura è opzionale.
3. ***Picture:*** definisce il tipo di frame ed il range del vettore del moto in quell'immagine.
4. ***Slice:*** L'unità di risincronizzazione, una collezione di macroblocchi è preceduta da un unico pattern di risincronizzazione (start code).
5. ***Macroblocco:*** L'unità di moto compensazione, una regione di 16x16 campioni luma combinata con due 8x8 campioni di entrambi i componenti chroma, l'header del macroblocco indica il metodo di predizione del blocco, l'indirizzo, coded block pattern e, se desiderato, può causare una variazione nel passo di quantizzazione rispetto a quello di base.
6. ***Blocco:*** l'unita di base per la trasformazione, una collezione di 8x8 pixel.

Uno slice consiste in due o più macroblocchi contigui lungo la stessa riga sono raggruppati insieme per formare una slice, l'ordine dei macroblocchi segue il tipico ordine di rasterizzazione delle televisioni (sinistra verso destra). Gli slice forniscono un efficiente meccanismo per limitare la propagazione degli errori, poiché un bitstream codificato consiste soprattutto in codeword a lunghezza variabile, ed ogni errore nella trasmissione può causare nel decoder la perdita dell'allineamento delle parole; ogni slice comincia con uno start code, e l'MPEG garantisce che una combinazione di codeword legali non può mai emulare uno start code, in questo

modo lo slice può essere utilizzato per riguadagnare il senso dell'allineamento delle codeword e riprendere la corretta decodifica.
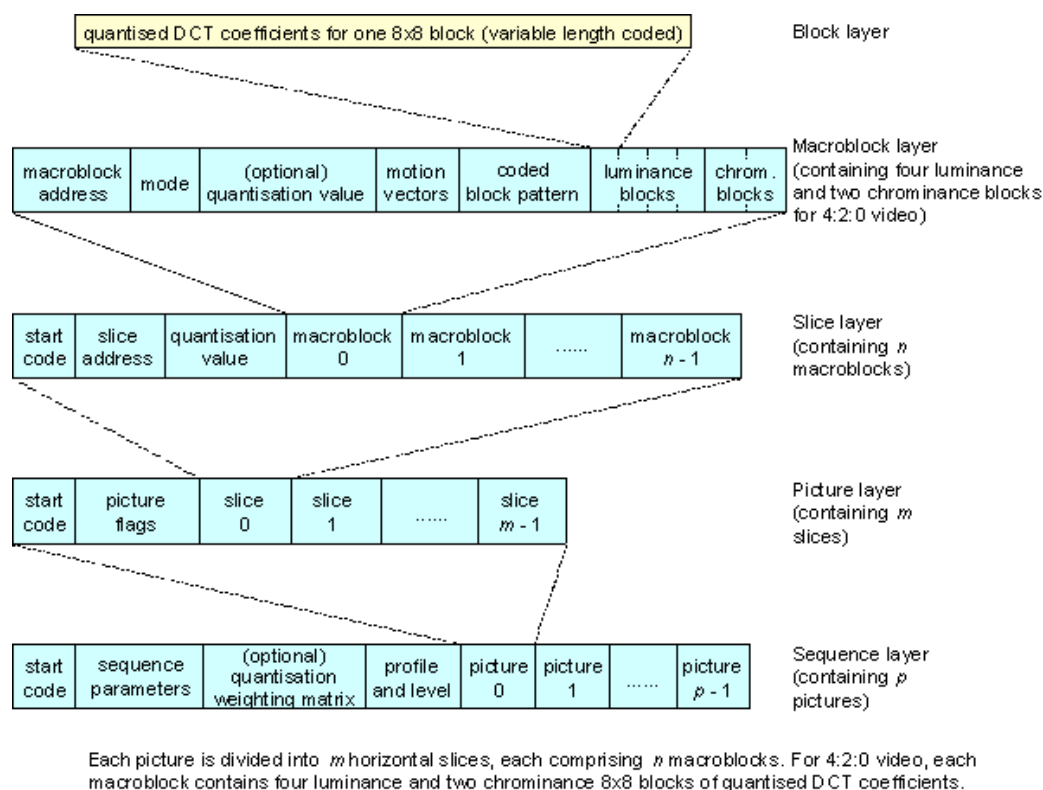
Immagine 2.16 - Layering del Frame MPEG-2

2.3.1.9 Buffer di Canale

Quando è utilizzata una codifica entropica il bitrate prodotto dal codificatore è variabile in funzione delle statistiche del video, poiché il bitrate permesso dai sistemi di trasmissione è minore del picco che può essere prodotto dalla codifica è necessario un buffer di canale nel decoder. Questo sistema di buffering deve essere progettato con attenzione, il controller deve permettere un efficiente allocazione dei bit ed assicurare che non avvengano situazioni di overflow o underflow. Il controllo del buffer tipicamente coinvolge meccanismi di feedback per l'algoritmo di compressione che riguardano la risoluzione di ampiezza (quantizzazione), e/o le risoluzioni spaziale, temporale e del colore che possono essere variate secondo le richieste del bitrate, aumentando o diminuendo l'efficienza della codifica e la qualità del segnale.

2.3.1.10 Livelli e Profili

L'MPEG-2 è disegnato per indirizzarsi ad un largo bacino di potenziali applicazioni, ed per essere efficiente in ciascuna ha un largo insieme di compliance point; essi sono indirizzati per livelli e profili:

- **profilo:** è un subset della sintassi del bit stream MPEG-2 standard con le restrizioni della parte dell'algoritmo MPEG utilizzato; sono analoghi alle features e descrivono le caratteristiche disponibili.
- **livello:** specifica i parametri generali come dimensione dell'immagine, data rate, grandezza del buffer del decoder; in essenza descrive il limite superiore per una data feature, è analogo alle specifiche della performance.

I livelli previsti sono:

- *Low (basso):* corrisponde alla risoluzione più bassa.
- *Main (principale):* corrisponde alla struttura 4:2:0 fino ad una risoluzione di 720 x 576 pixel.
- *High-1440 (alto-1440):* dedicato alla tv ad alta definizione HDTV.
- *High (alto):* ottimizzato per il formato di schermo 16/9 in alta definizione.

I Profili sono i seguenti:

- *Simple:* permette di semplificare notevolmente sia il codificatore di stazione che il decodificatore di utente in quanto non utilizza la predizione di tipo B.
- *Main:* è quello che offre il miglior compromesso tra qualità e tasso di compressione, impiega le immagini relative alle predizioni I, P, B a svantaggio dei dispositivi di codifica e decodifica che sono più complessi.
- *Scalable:* è destinato ad applicazioni particolari dove sia necessario ad esempio mantenere la compatibilità tra alta definizione e definizione standard oppure, riuscire ad ottenere una qualità accettabile in condizioni di ricezione difficile come potrebbe accadere ad esempio nella televisione digitale terrestre.
- *High:* è destinato all'alta definizione con le strutture 4:2:0 e 4:2:2.

I profili mantengono una certa compatibilità verso l'alto nel senso che, nella fase di ricezione, i profili più alti possono decodificare i profili inferiori.

2.3.2 MPEG Systems

I sistemi MPEG hanno i ruoli chiave del multiplexing e della sincronizzazione richiesta per provvedere ad una significativa ricostruzione del materiale multimediale. l'MPEG-2 sistem layer ha due versioni, una è il *program-stream*, ripresa dalla prima versione dello standard ottimizzata per i dispositivi di mantenimento, risul-

tante in lunghi pacchetti a dimensione-variabile (si utilizzano pacchetti lunghi perché si assume un basso error-rate), l'altra è il *transport-stream* che utilizza pacchetti piccoli a dimensione fissa per facilitare le operazioni di in caso di canali rumorosi e ci si aspettano operazioni di multiplexing/demultiplexing multicanale in molte operazioni broadcast. Entrambe le versioni condividono la filosofia di temporizzazione basata su comunicazioni di un comune riferimento temporale tra l'encoder ed il decoder, ed sul timestamping desiderato nella decodifica.

2.3.2.1 Timing

Il clock di sistema (o program-clock) è un riferimento di tempo comune mantenuto da encoder e decoder; nell'MPEG-2 consiste in un contatore a 33-bit incrementato da un contatore a 9-bit modulo 300 che lavora ad una frequenza di 27 MHz (ciò permette la retrocompatibilità dato che i 33 bit più significativi del contatore vengono aggiornati ad una frequenza effettiva di 90 kHz il che corrisponde esattamente al contatore dello standard MPEG-1). Il codificatore periodicamente campiona i valori del contatore e invia il valore corrente al decoder, il quale lo compara con con la sua versione ed eventualmente aumenta o diminuisce la sua frequenza a seconda del ritardo o dell'anticipo rispetto al timestamp ricevuto. Una volta stabilito il clock di sistema comune l'encoder codifica esplicitamente i valori del tempo a cui vuole che accadano i seguenti eventi chiave:

- *Presentation Time:* l'istante in cui un unità di accesso (un frame video o audio) deve essere riprodotto nel decoder specifico (target).
- *Decode Time:* l'istante in cui l'unità di accesso è codificata nel decoder specifico.

Data l'assunzione che il decode time è istantaneo i due differiscono esclusivamente nel caso della presenza di un buffer di riordino, ciò accade solo per le immagini ancora (frame-I o P) nel caso della presenza di frame-B nel video, nel caso di questi ultimi l'istante di decodifica e quello di riproduzione coincidono e dunque viene inviato solo il presentation time. Il decode time è cruciale per stabilire una delle condizioni chiave, quanto ritardare il l'inizio della decodifica finche non è ottenuto il mirror-state; in caso venga ignorato non c'è garanzia di evitare eventuali underflow o overflow che possano presentarsi in qualche punto del futuro. Il transport-stream dell'MPEG-2 possiede un'altra costante specifica nel program-clock che permette al decoder di bloccare la frequenza delle codifiche audio/video rispetto ai campioni del program-clock; questo evita la necessità di un addizionale *phase-locked-loop* basato sulla presentazione di timestamp in modo da recupera-

re ogni campione del clock; ciò inoltre impone un'alta aspettativa di fedeltà al program-clock. Il caso chiave in cui ciò può diventare un issue è quando l'arrivo di uno stream include riferimento al clock in anticipo od in ritardo rispetto a quanto originariamente intendessero, ciò può accadere durante il remultiplexing o l'adattamento del protocollo che non aggiustino esplicitamente il timestamp (come una trasmissione ATM). La variazione tra l'arrivo aspettato e quello effettivo è equivalente all'introduzione di una sorgente di rumore introdotta nel clock del decodificatore che risulta in variazioni della frequenza e della fase. In casi estremi può provocare variazioni nel timbro (*pitch*) audio e nella sottoportante dei colori.

2.3.2.2 System-Program Streams

Il system-stream consiste in una sequenza di pacchi ciascuno dei quali è una collezione di pacchetti che corrisponde ad uno stream elementare (audio, video o dati).
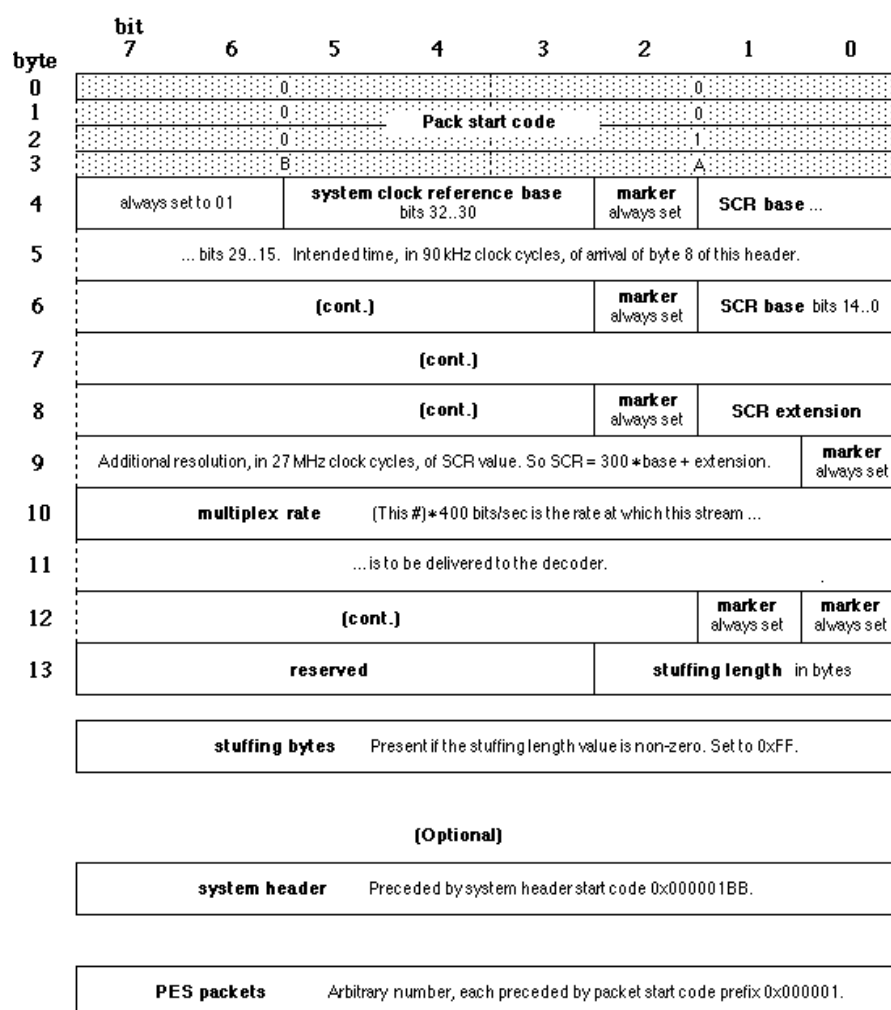
Immagine 2.17 - MPEG-2 Program Stream

L'header del pacco include un riferimento al clock ed un insieme di altri parametri; un pacco occasionalmente include l'header del sistema (in uno stream destinato allo storage il primo pacco generalmente contiene l'header di sistema, i pacchi successivi possono eventualmente contenere variazioni dei parametri del sistema), esso contiene informazioni richieste alla configurazione del decoder, esse includono il numero ed i tipi di streams, le relazioni tra i clock audio, video e di sistema e la dimensione del buffer di decodifica richiesto.

2.3.2.3 Transport Streams

Un transport stream è composto di pacchetti dalla dimensione fissa di 188-byte, che includono sempre un byte di sincronizzazione di un valore fissato (47 in esadecimale); poiché questo valore può apparire anche tra i dati l'acquisizione del pacchetto consiste nel cercare per primo il byte di sincronizzazione per poi verificare che si ripresenti 188 bytes dopo per un certo numero di volte. Il restante header del pacchetto include:

1. un bit indicatore di errore per segnalare al decoder di intramprendere un'azione di mascheramento dato che il pacchetto è stato danneggiato durante la trasmissione.
2. l'indicatore dell'unità payload di inizio che fornisce un segnale facilmente estraibile dal decoder per capire se un pacchetto PES o un altro payload stà iniziando nel pacchetto corrente, ciò facilità lo start-up del decoder.
3. Transport priority è un singolo bit che indica se il pacchetto corrente è a bassa o alta priorità.
4. PID il packet ID è un indirizzo di 13 bit che indica quale stream elementare è trasportato dal pacchetto (o alternativamente quale tipo di tabella o stream privato è trasportato).
5. Transport stream scrambling control è presente (come altre feature del pacchetto) per facilitare la sicurezza della trasmissione senza definire formalmente un protocollo.

Occasionalmente i restanti 184 byte del pacchetto includono un campo di adattamento il cui contenuto di maggiore importanza è il riferimento al program-clock, può anche includere indicatori sulle discontinuità e come splice gli stream, dati privati, strumenti per l'aggiustamento del ratee del pacchetto e altro.
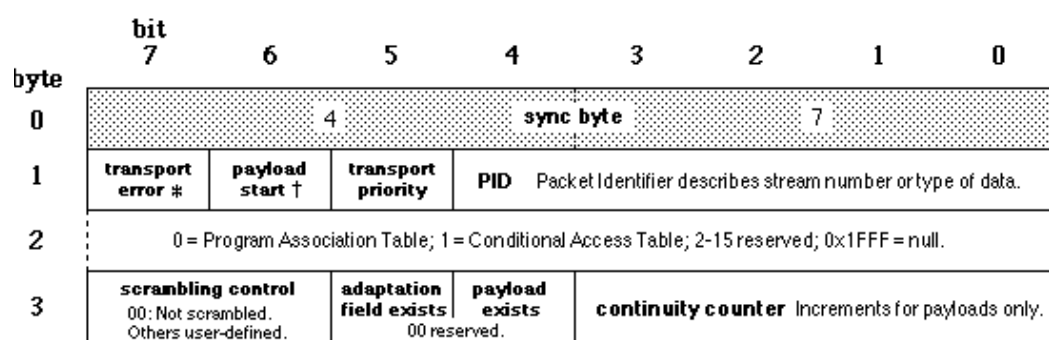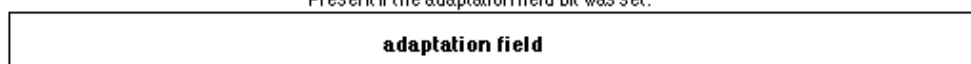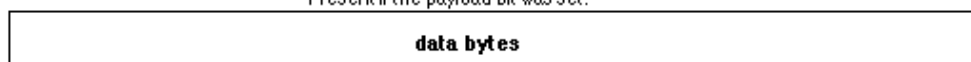
Immagine 2.18 - MPEG-2 Transport Stream

2.3.2.4 Packetiszed Elementary Streams (PES)

Sia i Transport-Streams che i Pregram-Streams trasportano dei sottelementi comuni, i PES, ciascun PES contiene informazioni su un unico stream elementare.

Immagine 2.19 - MPEG-2 PES

Il pacchetto consiste in un header che include un start-cede-prefix e uno stream-ID che identifica lo stream elementare trasportato, in particolare 32 per l'audio e 16 per il video. Nel caso dei program-streams è utilizzato per determinare a che tipo di decoder audio o video inviare il contenuto del pacchetto; nel caso del transport-stream l'informazione è ridondante poichè solo i pacchetti PES provenienti da un singolo stream possono essere trasportati da un transport-stream con un dato PID. L'header contiene anche la dimensione del pacchetto ed un certo numero di informazioni opzionali, la più importante delle quali è il PST (presentation time stamp) che può essere presente se un frame audio o video ha un data start compresso nel pacchetto; ed un DTS (decode time stamp) se il PTS è presente ed i due sono diversi.

2.3.2.5 Program-Specific Information

L'MPEG-2 definisce un insieme di chiavi per facilitare l'associazione di streams elementari audio e video in collezioni chiamate "programmi"; un tipico programma consiste in uno stream video, uno o più streams audio (diversi linguaggi) e zero o più stream dati (sottotitoli o messaggi di sicurezza) La tabella di associazioni del programma è sempre portata nel PID 0 in un transport-stream, per ogni programma contenuto nel flusso determina la locazione, tramite PID della tabella che descrive gli streams associati con quel programma (Program Map Table), indicandone la locazione e la caratteristiche individuali; inoltre provvede il PID di una speciale tabella in cui sono contenute le informazioni sul transport-stream corrente e su altri che possono essere disponibili sulla stessa rete.

2.3.3 MPEG-4

Lo standard ISO/IEC 14496 parte 2 (MPEG-4 Visual) migliora il popolare standard MPEG-2 in termini di efficienza nella compressione (migliore riduzione dei dati con medesima qualità) e flessibilità (permette un maggior numero di applicazioni); questi obiettivi sono raggiunti facendo uso algoritmi di compressione più avanzati e fornendo un esteso insieme di strumenti per la manipolazione e la codifica dei video digitali. Il nucleo intorno al quale è costruito l'MPEG-4 è un algoritmo DCT con compensazione del moto con alcune estensioni rispetto all'MPEG-1 e 2; in particolare permette una compensazione del moto più specifica sui contenuti; includendo le stime del moto standard basate su blocchi 8x8 o 16x16, compensazione del moto globale basata su otto parametri di moto che descrivono una trasformazione affine, o la compensazione del moto di "sprites" (uno sprite statico,

così come una larga immagine fissa che descrive uno sfondo panoramico, moto compensato da immagine ad immagine, ed un insieme di sprites dinamici generati attraverso la scena ed in seguito compensati). Oltre ciò si aggiungono strumenti per una codifica efficiente di VO (*Video Object*) al di là dell'immagine, inclusi strumenti per la compressione delle textures, ed il texture mapping su meshes 2D o 3D, compressione di meshes 2D implicite e compressione di flussi geometrici tempo-varianti che descrivono l'animazione di una meshes. Inoltre è data importanza ad algoritmi scalabili che permettono la decodifica di solo parti dello stream in modo da ricostruire la sequenza di immagini ad una ridotta risoluzione spaziale, temporale o qualità. Infine provvede delle facilitazioni nell'integrazione di oggetti visuali all'interno della scena basate su un modello gerarchico. Al livello più alto si definisce un sistema di coordinate globale (nello spazio e nel tempo), ogni figlio ha un sistema di coordinate locali ed un offset rispetto alle coordinate paterne che ne permette la localizzazione, i nodi di quest'albero di descrizione possono essere aggiunti e rimossi dinamicamente ed i loro attributi (come la posizione rispetto alle coordinate del padre) possono essere modificati; l'abilità di modificare elementi nella descrizione della scena permette l'interazione dell'utente con gli oggetti visuali. Questo standard è stato sviluppato in un periodo in cui c'era una particolare necessità di una rappresentazione di contenuti multimediali adatta alla reti di calcolatori, in modo particolare Internet; queste reti sono caratterizzate da un alto grado di variabilità e flessività nella capacità computazionale dei terminali e nelle caratteristiche della trasmissione, include banda e specifiche di qualità del servizio. L'MPEG-4 abbraccia varie aree:

1. Codifica di Oggetti Visuali: estensione dagli standard precedenti, per includere una maggiore flessibilità negli algoritmi classici basati su pixel per estenderli ad applicazioni a bitrate minore, ed aggiunta di strumenti per la composizione ed il rendering di altri elementi grafici diversi dai pixel.
2. Codifica di Oggetti Audio: esteso per supportare algoritmi di compressione audio per bitrate minori (generalmente specifici per il parlato) e per rendere possibile la composizione di suoni ed effetti sintetizzati.
3. Descrizione della Scena ed Interazione con l'utente: strumenti per la descrizione gerarchica di elementi della scena che sono combinati per generare l'immagine finale (potenzialmente modificabile dall'utente).

4. Sistemi: protocolli per handling demultiplexing, gestione del buffer e temporizzazione, un protocollo sofisticato per permettere interfaccia e controllo trasparenti tra le applicazioni MPEG-4 distribuite su una gran varietà di reti di trasporto.

L'MPEG-4 fornisce anche strumenti per la sintesi sia di audio che di video, utilizzando descrizioni parametriche, può combinare questi elementi sintetici con elementi naturali in una scena comune; ci sono anche strumenti per l'interattività che permettono all'utente di modificare gli elementi della scena. Per l'obiettivo di questa tesi ci concentreremo solo sugli aspetti di compressione e rappresentazione di una scena naturale.

2.3.3.1 Video Object

L'MPEG-4 non tratta la sequenza video nel senso tradizionale di una specifica sequenza di frame video rettangolari; al contrario considera un video come una collezione dinamica di uno o più oggetti video; un *Oggetto Video* (VO o *Video Object*) è definito come un entità flessibile che a cui l'utente è permesso di accedere e manipolare. Più praticamente un VO è un area della scena video che occupa una regione di forma arbitraria è può esistere per un arbitrario periodo di tempo; un istanza di un oggetto video in un particolare punto del tempo è un VOP (*video object plane*). Questa definizione comprende l'approccio tradizionale di codifica dell'intero frame, visto come un singolo VOP e la sequenza un singolo VO; tuttavia permette un approccio più flessibile per la codifica. Ad esempio una scena video può essere suddivisa in un oggetto video che contiene lo sfondo, ed una serie di VO ciascuno dei quali rappresenta un separato oggetto in primo piano. Questo approccio aumenta la flessibilità dell'intero sistema, ciascun oggetto può essere codificato separatamente diversificandone la qualità e la risoluzione spaziale o temporale. Nonostante le potenzialità offerte da una codifica basata sugli oggetti le applicazioni più popolari dell'MPEG-4 continuano a servirsi della codifica di un intero frame video. L'MPEG-4 riprende gli strumenti utilizzati per la compressione audio e video dell'MPEG-2 estendendoli il concetto di frame video in quello di un oggetto video di forma arbitraria; perciò i Frame-I, P o B diventano I-VOP, P-VOP e B-VOP mentre i gruppi di immagini (GOP) si trasformano in gruppi di piani video (GOV).
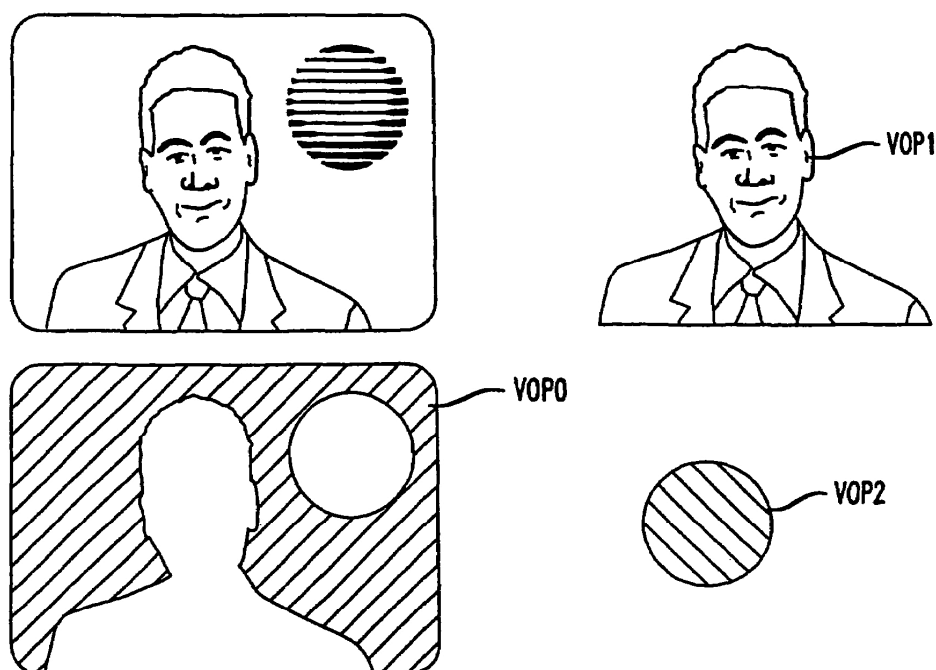
Immagine 2.20 - Video objects

2.3.3.2 Codifica della Forma

Un video di forma rettangolare (chiamato texture) è codificato in modo molto simile all'MPEG-2 eccetto che le dimensioni del rettangolo possono essere scelte arbitrariamente, il modo in cui degli oggetti video di forma arbitraria sono codificati fa uso del canale alpha, il vero oggetto video è definito in un area rettangolare, chiamata maschera, dimensionata opportunamente per aggiustare la massima estensione orizzontale e verticale del video al più vicino multiplo di 15 pixel. Il canale alpha indica se e quanto i pixel sottostanti sono visibili, le forme possono essere codificate come dati binari o a livelli di grigio, il primo metodo è il più semplice con il canale alpha che controlla semplicemente se un pixel è visibile o trasparente. Sfortunatamente questo conduce ad un edge effect che viola il teorema di Nyquist e può condurre a visibili ed indesiderati artefatti di aliasing lungo i bordi. Una forma codificata con livelli di grigio permette una transizione più dolce e controllata tra l'oggetto e lo sfondo. Queste tecniche di alpha blending rendono l'apparenza di un oggetto video arbitrario più realistiche. In ogni caso i pixel di contorno che rappresentano i bordi dell'oggetto sono codificati utilizzando tecniche di codifica aritmetica e tecniche entropiche.

2.3.3.3 Codifica delle Texture

La codifica delle texture è l'analogo della codifica convenzionale delle immagini in movimento nell'MPEG-2, in tutti i profili dell'MPEG-4 ad eccezione di quello studio

è utilizzata un color subsample di 4:2:0 per descrivere una texture, nonostante l'area rettangolare che rappresenta l'estensione dell'oggetto video solo i macro-blocchi resi visibili dal segnale dell'immagine, cioè quelli totalmente o parzialmente all'interno dei confini della forma dell'oggetto, sono effettivamente codificati. gli I-VOP sono codificati in maniera molto simile ai frame-I, con una maggiore efficienza dovuta all'uso di un migliore predittore, esso misura i gradienti delle luminosità orizzontale e verticale e predice il coefficiente DC dal blocco superiore ed a sinistra nella direzione del minimo gradiente.

2.3.3.4 Codifica dei Bordi

Quando i blocchi fanno parte del confine arbitrario di un oggetto nasce un interessante problema, i pixel che giacciono al di fuori del bordo dell'immagine devono avere un certo valore, ma idealmente la scelta di questi valori, che non contribuiscono all'immagine visibile, non dovrebbe aggiungere energia ai coefficienti del blocco o ci sarebbero effetti di marked nel processo di DCT, scegliere di rendere questi pixel neri ad esempio sarebbe una pessima mossa. Per evitare il problema l'MPEG-4 assegna ai pixel che non fanno parte dell'immagine un valore pari alla media di tutti i pixel che fanno parte dell'immagine, questo processo minimizza l'energia dei coefficienti quando il blocco è trasformato tramite DCT.

2.3.3.5 Codifica di Oggetti Video di Forma Arbitraria

Con oggetti video di forma arbitraria i blocchi possono essere totalmente trasparenti, totalmente opachi o far parte del bordo del VOP, nel primo caso non è richiesta alcuna texture, nel secondo la texture può essere intra-codificata o predetta con compensazione del moto come un macroblocco dell'MPEG-2. Per i blocchi del terzo tipo è necessario codificare sia la texture (immagine video) che la forma dell'oggetto (canale alpha) ed entrambe possono essere intra-codificate o predette.

2.3.3.6 Sprites

L'MPEG-4 ha un altro interessante tipi di oggetto video, utile per gli sfondi, utilizzato particolarmente nelle applicazioni interative e consiste in un intero sfondo la cui visione è solo parziale e dipende dal punto di vista ad un determinato istante.

2.3.3.7 Advanced Coding Extensions (ACE)

L'MPEG-4 introduce diversi strumenti che migliorano l'efficienza della codifica per gli oggetti video.

1. *4 Motion Vector per Macroblock:* Come già detto la predizione del moto tende ad essere più efficace se la dimensione del blocco è minore, lo standard defi-

nisce il macroblocco di base come una regione di 16x16 pixel più due di 8x8 che vengono predette da un singolo vettore del moto. L'MPEG-4 include l'opzione che permette di suddividere in quattro questa regione, creando dei sottoblocchi di 8x8 campioni di luma e 4x4 di chroma, ed ottenendo quattro vettori per macroblocco. Questo metodo permette una maggiore efficienza nella riduzione dell'energia ma introduce un overhead, il codificatore può decidere sul momento se utilizzare questo metodo o il classico per incrementare l'efficienza globale.

2. *Unresricted Motion Vector:* in alcuni casi il best-match per un macroblocco può essere una regione di che si estende al di fuori dei limiti del VOP di riferimento, questo strumento permette al vettore del moto di puntare oltre i limiti dell'oggetto, e ciò migliora l'efficienza, specialmente nel caso di oggetti che entrano ed escono dall'immagine.
3. *Intra Prediction:* i coefficienti della trasformazione che indicano le base frequenze in blocchi 8x8 adiacenti sono spesso correlate, in questo modo il coefficiente DC ed eventualmente i coefficienti AC (i coefficienti che si trovano sulla prima riga e colonna della matrice) possono essere predetti dai blocchi adiacenti già codificati, generalmente il superiore e il precedente.
4. *Global Motion Compensation:* permette al movimento globale dell'intero VO di essere codificato con soli pochi parametri, se l'oggetto rimane stazionario o ha un movimento non deformante non c'è necessità di indicare il movimento di ogni singolo blocco di cui è composto.
5. *Quarter Per Motion Compensation:* La risoluzione migliorata per i vettori del moto sostanzialmente riduce l'errore di predizione e la conseguente necessità di trasmettere i residui.
6. *Shape Adaptive DCT:* può migliorare la codifica nei blocchi di confine, al posto di applicare semplicemente la DCT ad un intero blocco viene applicata la DCT monodimensionale prima verticalmente, poi orizzontalmente, ma solo ai pixel effettivamente visibili (pixel attivi). Per iniziare ciascun blocco è esaminato per cercare pixel attivi, per ogni colonna cha ne contiene questi pixel sono *top justified* cioè mossi in cima alla colonna ed è eseguita una DCT di dimansioni 1x# di pixel attivi; quando le trasformazioni verticali sono completate il processo viene ripetuto per ciascuna riga; il risultato è un minor numero di coefficienti da essere trasmessi; poichè il contorno dell'oggetto è trasmesso separatamente

dal codice di shaping i pixel possono essere facilmente ripristinati in fase di decodifica.

2.3.3.8 Audio Object

Gli oggetti audio sono definiti per diversi tipi di materiali, il suono naturale può essere compresso a bitrates di 2-64 kbps, sono utilizzati tre tipi di algoritmi di compressione per suddividere questo range: il range 2-6 kbps utilizza un a codifica parametrica per il parlato con un campionamento ad 8 kHz, per il range 6-24 kbps utilizza un predittore lineare pilotato da codici ottimizzato per il parlato a frequenze di campionamento di 8 o 16 kHz, oltre i 16 kbps si una l'algoritmo di compressione AAC studiati apositamente per l'alta qualità. Oltre la compressione di suoni naturali lo standard possiede strumenti per la rappresentazione di suon sintetizzato.

2.3.3.9 Multiplexing e Trasporto

Gli oggetti audio-video dell'MPEG-4 sono trasmessi in flussi elementari che includono informazioni riguardo a parametri, come le risorse richieste dal decoder (dimensione dell'immagine, dimensione del buffer) e l'aspettativa riguardo la qualità del servizio che deve essere mantenuta durante la trasmissione. L'Access Unit Layer raccoglie informazioni da ognuno degli streams elementari in elementi liberamente indirizzabili, come frame video. Due Layer del multiplexing sono applicati all'unità di accesso, il primo, un multiplexing interno chiamato FlexMux, permette il raggruppamento di unità d'accesso con basso overhead e caratteristiche simili; il secondo, multiplexing di trasporto o TransMux, è un meccanismo di trasporto definito esternamente all'MPEG-4, se il TransMux provvede tutte le funzionalità necessarie, allora il FlexMux non è necessario, esempi di Layer TransMux sono RTP/UDP/IP o il transport-stream dell'MPEG-2. Oltre al meccanismo di multiplexing nell'MPEG-4 è presente il *delivery multimedia integration framework* (DMIF), un interfaccia trasparente per l'interazione con con i nodi (peers) MPEG-4, indipendentemente dalla loro collocazione, il DMIF supporta nodi multipli interattivi (capaci di reagire a segnali da altri peers) o meno; esso può essere responsabile per l'inizializzazione del canale con la qualità del servizio e la banda desiderate, e modificare queste proprietà dinamicamente durante la sessione.

2.3.3.10 Error Robustness

L'MPEG-4 si indirizza anche ad applicazione che fanno spesso uso di canali ad elevato tasso di errori, per questo motivo lo standard fornisce strumenti per migliorare la robustezza agli errori.

1. *Resincronizzazione:* cerca di permettere la resincronizzazione tra il decoder ed il bitstream dopo che è stato scoperto un errore residuo, in generale i dati tra il precedente punto di sincronizzazione ed il punto in cui essa è ristabilita vengono eliminati. L'MPEG-2 che fa uso di slice (gruppi di blocchi) a cui vengono assegnati degli start code univoci, questo approccio posiziona i marker di sincronizzazione in dipendenza dei blocchi, e con codifiche a lunghezza variabile ciò si traduce in un posizionamento non prevedibile degli start code lungo il bitstream, come conseguenza aree con elevato tasso di movimento tenderanno saranno più soggette a perdita dei dati. l'MPEG-4 adotta un approccio che prevede la presenza di marker di sincronizzazione periodici lungo il bitstream, la posizione del marker non dipende dal numero di blocchi ma dal numero di bit contenuti nel pacchetto; il marker è distinto da qualsiasi codeword VLC così come dallo start code del VOP, l'header del pacchetto contiene tutte le informazioni necessarie per la sincronizzazione. L'MPEG-4 adotta anche un altro approccio chiamato *fixed interval synchronization*, questo metodo richiede che lo start code del VOP e quello del pacchetto video (marker di resincronizzazione) siano posizionati ad intervalli prefissati nel bitstream, ciò previene la il decoder dall'interpretare una sequenza corrotta come uno start code legale, poiché deve cercarli sono all'inizio di ogni intervallo.
2. *Data Recovery:* dopo che la sincronizzazione è stata ristabilita è necessario recuperare i dati che altrimenti andrebbero persi, l'MPEG-4 fornisce strumenti che permettono di codificare i dati in modi fondamentalmente resistenti agli errori; uno di questi strumenti è il RVLC (*reversible veriable lenght code*) che consiste nel progettare i codewords di lunghezza variabile in modo che possano esser eletti in entrambe le direzioni, in pratica il marker di sincronizzazione può essere utilizzato come uno starting point per leggere il bitstream in ordine inverso, fino al punto in cui si è verificato l'errore che ha causato la perdita di sincronizzazione. senza questo approccio tutti i dati dalla presenza dell'errore in poi sarebbero stati scartati, anche se validi, semplicemente perché il decoder non era in grado di riconoscerli come tali.
3. *Error Concealement:* questo strumento agisce come l'ultima linea di difesa in un codificatore robusto e tollerante agli errori; l'efficienza di questa strategia dipende ampiamente dalle performance dello schema di resincronizzazione, se riesce a isolare efficacemente l'errore nel flusso allora il problema del masche-

ramento dell'errore diventa più facilmente trattabile, in applicazioni a basso bitrate il semplice espediente di copiare blocchi dal VOP precedente può fornire dei risultati adeguati. Per migliori strategie l'MPEG-4 fornisce un metodo che migliora ulteriormente le capacità di localizzazione dell'errore, questo approccio si serve del partizionamento tra il moto e le texture, ed un secondo marker di sincronizzazione è inserito tra le due, perciò se viene smarrita l'informazione relativa all'immagine si può utilizzare la compensazione del moto per mascherare l'errore.

2.3.3.11 Strumenti, oggetti, profili e livelli

L'MPEG-4 fornisce le sue funzioni di codifica tramite una combinazione strumenti, oggetti e profili. Uno strumento è un sottoinsieme delle funzioni di codifica per il supporto di una specifica caratteristica; un oggetto è un elemento video che è codificato da uno o più strumenti; un profilo è un insieme di tipi di oggetti che il CODEC è capace di gestire. Nell'MPEG-4 sono presenti un totale di 34 diversi profili, suddivisi a seconda del target di codifica (video naturali, video sintetizzati, video ibridi, computer grafica, ecc...), nel nostro lavoro utilizziamo due codec che si servono dell'Advanced Simple Profile, le cui caratteristiche comprendono la codifica di video rettangolari con efficienza migliorata e supporto per video interlacciato. I profili definiscono un sottoinsieme di strumenti di codifica e Livelli definiscono i parametri costanti della codifica; l'ASP definisce sei livelli.

Livello	Tipica Risoluzione	Max bitrate	Max Oggetti
L0	176x144	128 kbps	1 AS o Semplice
L1	176x144	128 kbps	4 AS o Semplice
L2	352x288	384 kbps	4 AS o Semplice
L3	352x288	768 kbps	4 AS o Semplice
L4	576x352	3 Mbps	4 AS o Semplice
L5	720x576	8 Mbps	4 AS o Semplice

2.4 Compressione Audio

Un segnale audio complesso contiene una grande quantità di informazione, molta della quale, poiché non può essere percepita dall'orecchio umano, è considerata irrilevante, lo stesso segnale, e seconda della sua complessità, può contenere informazioni che sono altamente prevedibili e dunque, ridondanti. La ridondanza è

misurabile e quantificabile e può essere eliminata dal codificatore per poi essere ripristinata dal decodificatore, questo processo è spesso definito compressione statica; l'irrilevanza, d'altro canto, è riferita come una codifica percettiva, una volta eliminata non può essere ricostruita, questo procedimento è tipicamente soggettivo ed è sviluppato da ogni algoritmo secondo un diverso modello psicoacustico.

2.4.1 Audio MPEG ed MP3

Lo standard di compressione audio MPEG definisce una serie di algoritmi appropriati per un ampia gamma di materiale audio, basata su una combinazione di compressione di sottobande con tecniche che esplicitano le possibilità uditive dell'orecchio umano.

2.4.1.1 Layers

Come per il video anche lo standard audio si indirizza a diversi punti di compliance, tali livelli nell'MPEG audio sono chiamati Layer, ed ordinati da uno a tre al crescere della complessità, per il tipico materiale musicale una coppia stereo può essere codificata con circa 64 kbps con il layer 3, 128 kbps con il 2 e 192 kbps con l'uno. Tra di essi il l'algoritmo più famoso ed ampiamente utilizzato è l'MPEG-1 layer 3, più noto con l'acronimo MP3; è in grado di ridurre drasticamente la quantità di dati richiesti per riprodurre un suono, rimanendo comunque una riproduzione fedele del file originale non compresso.

2.4.1.2 Algoritmo di base

La decomposizione delle sottobande separa il segnale in arrivo in multiple bande di frequenza, che sono in seguito scalate (tramite uno *slowly scaling factor* per ogni sottobanda) e quantizzate, inoltre per ogni sottobanda è definita un'addizionale analisi nel dominio della frequenza per selezionare il passo di quantizzazione adatto. i campioni sono codificati (con lungheza fissa nei primi due layer e con lunghezza variabile nel 3) e formattati con informazioni di supporto. Il decoder scompatta i coefficienti e le informazioni di supporto, esegue la quantizzazione inversa per mappare ogni campione nel uo valore ricostruito, moltiplica per l'appropriato fattore di scala ed applica la composizione delle sottobande per ricostruire il segnale nel dominio del tempo.

2.4.1.3 Decomposizone in sottobande

La decomposizione è eseguita utilizzando un filtro a quadratura a 32 bande (QMF quadrature mirror filter) per il layer 3 a ciò è seguita una DCT modificata, la combinazione porta ad una più fine risoluzione della frequenza, ottenendo un massimo

576 bande nel bank del filtro che può essere switchato tra l'analisi a long-time (risoluzione ad alte frequenze) o short-time (risoluzione a base frequenze) in presenza di forti variazioni temporali. Questa scelta è disponibile sulla base della distorsione che avviene quando un improvviso cambio nel contenuto (attacco) è codificato e compresso con una finestra long-time, risultando in una distorsione che precede e segue l'attacco di un ampia misura. Per tutti i layers la decomposizione in sottobande è codificata criticamente, il numero di campioni per unità di tempo in ciascuna sottobanda per il numero di sottobande è uguale al numero di campioni utilizzati per unità di tempo nel dominio originale.

2.4.1.4 Scalatura, Quantizzazione e codifica

Per ogni sottobanda un frame consiste in 36 consecutivi campioni, durante la decodifica un fattore di scala (che agisce come un guadagno) moltiplica il campione, si può aplicare lo stesso fattore per tutti i campioni della sottobanda o variare il guadagno per ogni gruppo di 12 campioni successivi, separando in questo formato di guadagno/ampiezza la quantizzazione può focalizzarsi sulla più narrow range dinamico dei campioni dopo la divisione per il fattore di scala, questa strategia è implicitamente allineata con il concetto di mascheramento audio, secondo cui distorsione più ampie sono accettabili in bande di frequenza con un segnale ad energia più elevata; il modello guadagno/ampiezza produce un minore rumore di quantizzazione (poiché il rumore è moltiplicato dal fattore di scala quanto il segnale) in bande con bassa energia in input. Durante la codifica è inoltre applicata un esplicita analisi nel dominio della frequenza, questa analisi è utilizzata per guidare il modello psicoacustico sulla quantità di mascheramento possibile in ogni sottobanda, producendo un rapporto tra il segnale ed il mascheramento (signal-to-mask-ratio SMR), il rapporto segnale rumore sotto il quale ci si aspetto che il mascheramento prevenga ogni udibile distorsione. Sono inoltre allocati dei bits tra le sottobande in modo da minimizzare il totale della differenza tra il rapporto segnale/rumore in ogni banda che risulti dopo la quantizzazione ed il SMR in quella banda. L'utilizzo dell'appropriato modello psicoacustico rappresenta un sostanziale contributo alla performance del codificatore, modelli più sofisticati generalmente richiedono una più elevata risoluzione frequenziale che è fornita dallo stesso filtro delle sottobande (che è ottimizzato in finestre short-time per modificare pre-echi, la diffusione delle distorsioni che precedono un attacco); usando il numero di bit allocati

per campione in una sottobanda è applicato un pasos di quantizzazione uniforme per ogni valore di ampiezza.

2.4.1.5 Compressione Multicanale

L'MPEG a partire dal primo standard supporta esplicitamente la compressione stereo, ci sono quattro modi per indirizzare un canale singolo o doppio: single_channel (mono), dual_channel (una coppia scorrelata di canali mono), stereo (due canali intesi come una coppia stereo ma senza una speciale compressione) e joint_stereo; l'ultimo metodo utilizza una specifica caratteristica della percezione stereofonica umana, la localizzazione spaziale della sorgente del suono dipende sia dall'ampiezza che dalla fase del per le basse frequenze, ma solo dall'ampiezza per le alte, in altre parole è sufficiente avere un corretto rapporto loudness tra l'orecchio destro ed il sinistro per le alte frequenze così da apparire ad una locazione spaziale specifica senza doversi preoccupare della fase del segnale; ma questo non è sufficiente alle basse frequenze. La codifica joint-stereo permette al codificatore di specificare una sottobanda sopra la quale (in frequenza) i campioni dal segnale destro e sinistro hanno il fattore di scala calcolato indipendentemente (così che l'ampiezza in output possa essere mantenuta), ma il valore dopo la divisione per in fattore di scala sono mediate; queste medie sono ciò che viene effettivamente trasmesso ed utilizzato per ricostruire i segnali destro e sinistro, questo distrugge l'informazione della fase relativa tra i due canali; al di sotto della sottobanda specificata la compressione avviene normalmente, al di sopra questo metodo produce una guadagno nella compressione di circa il 5-10% nelle applicazioni tipiche, tuttavia non è compatibile con i sistemi che contano sull'informazione relativa alla fase del segnale per la ricostruzione del segnale stereo.

2.4.2 Audio AC3

L'AC3 o Dolby Digital è un sistema di codifica audio multicanale sviluppato da Dolby Laboratories Inc ed utilizzato al cinema, nella TV digitale, nei Laser Disc, DVD ed in altri supporti di riproduzione o trasmissione audio digitale. Il Dolby Digital è un sistema di codifica cosiddetto "lossy", ovvero in cui la codifica audio avviene con perdita di informazioni. Il sistema prende in input segnali audio digitali PCM (*Pulse-Code Modulation* è un metodo di rappresentazione digitale di una segnale analogico dove l'ampiezza del segnale è campionata ad intervalli di tempo regolari e quantizzata in una serie di codici binari, generalmente utilizzata nei sistemi di telefonia digitale), codificati a 48.000 Hz di frequenza di campionamento e

16 bit di risoluzione, li trasforma analizzandoli nel dominio della frequenza, per poi scartare parte dei dati riducendone la risoluzione effettiva ed innalzando il livello del rumore digitale. Tutta l'operazione viene eseguita cercando il più possibile di mantenere la soglia del rumore (di quantizzazione) al di sotto del livello udibile, utilizzando metodi di analisi di tipo "psicoacustico".

2.4.2.1 Codifica AC3

Uno stream AC3 è costruito da una serie di frame di sincronizzazione, composto nell'ordine di un serie di bit che contengono l'informazione di sincronizzazione (SI), un header di informazioni sul bitstream (BSI) lo segue, contenente i parametri del servizio di codifica audio; subito dopo è contenuta l'informazione audio, suddivisa in sei blocchi, ciascuno dei quali contiene 256 campioni audio per canale. Alla fine del bitstream sono inserite delle informazione ausiliarie ed
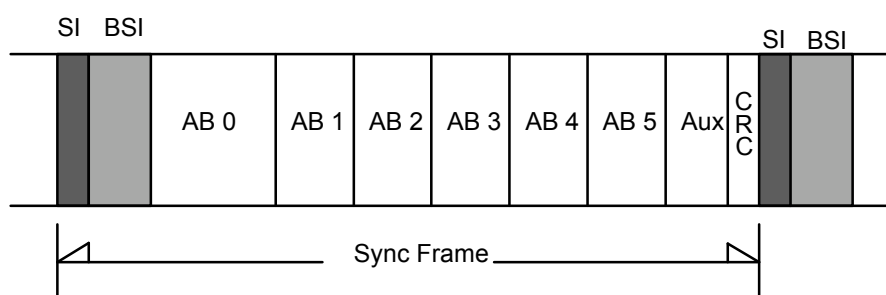
Immagine 2.21 - Frame AC3

Di seguito è un header AC3 semplificato che permette di avere un idea della sintassi del bitstream:

Field Name	# of bits	Description
syncword	16	0x0B77 Trasmessi con il metodo Big Endian.
CRC	16	
Sampling frequency	2	'11'=riservato '10'=32 kHz '01'=44.1 kHz '00'=48 kHz
Frame Size Code	6	
Bit Stream Identification	5	
Bit Stream Mode	3	'000'=servizio audio principale

Field Name	# of bits	Description
Audio Coding Mode	3	'010'=left, ordinamento del canale destro
Center Mix level	2	
Surround Mix Level	2	
Dolby Surround Mode	2	'00'=non indicato '01'=Surround encoded disattivato '10'=Surround encoded attivato

L'encoder accetta in input sequenza audio PCM di 1536 campioni, ciò è richiesto dal vincolo di mantenere lo stesso campionamento rispetto all'output; i canali di input individuali possono essere filtrati (filtro passa-alto).

2.4.2.1.1 Transient detection

I transitori (in acustica è un segnale di corta durata che rappresenta un attacco non armonico in un suono; contiene un elevato grado di componenti non periodici ed un'elevata ampiezza di alte frequenze) sono individuati nei canali ad ampiezza di banda piena per determinare quando passare da un blocco di trasformazione lungo (512) con un blocco corto (256), operando su 512 campioni per blocco audio suddividendo il procedimento in due passaggi da 256 campioni. L'individuazione dei transitori si suddivide in 4 passi:

Filtraggio passa-alto: il filtro è implementato come un filtro II IIR a cascata con un taglio a 8 kHz

Segmentazione del blocco in sottocampioni: ogni blocco è segmentato in un albero gerarchico di tre livelli. 1°livello l'intero blocco, 2°livello due segmenti di 128 campioni ed il 3° quattro segmenti da 64.

Individuazione dei picchi di ampiezza in ogni sottoblocco: individuazione del picco di ampiezza per ogni sottoblocco di ogni livello.

Comparazione delle soglie: per prima cosa verifica se c'è un livello significativo nel blocco corrente, in seguito verifica che il picco relativo a segmenti adiacenti in ciascun livello dell'albero, se il picco di due segmenti adiacenti eccede la soglia predefinita si setta un flag che indica la presenza di un transitorio.

2.4.2.1.2 Forward Transform

Il blocco audio è moltiplicato per una funzione finestra che migliora la selezione delle frequenze, ciascun blocco audio è trasformato nel dominio della frequenza eseguendo una trasformazione:

$$X_D[k] = \frac{-2}{N} \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi}{4N}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1)(1+\alpha) \right) \quad \text{for } 0 \le k < N/2$$

$$\text{where } \alpha = \begin{cases} -1 & \text{for the first short transform} \\ 0 & \text{for the long transform} \\ +1 & \text{for the second short transform} \end{cases}$$

2.4.2.1.3 Strategie di accoppiamento

Se abilitato l'accoppiamento dei canali viene realizzato mediando i coefficienti dei due canali inclusi nell'accoppiamento, ciascun canale ha un unico insieme di coordinate che preservano le alte frequenze che definiscono il segnale. Per un codificatore basilare può essere utilizzata una strategia di accoppiamento statica, utilizza dei parametri costanti inseriti direttamente nel bitstream, essi sono la frequenza iniziale e finale (indicati con il numero di sottobanda) e la struttura dell'accoppiamento delle bande; in alternativa è possibile utilizzare una strategia dinamica, in tal caso le frequenze di accoppiamento sono determinate da un modello psicoacuistico che compara l'udibilità degli artefatti causati dalla starvation rispetto a quelli creati dall'accoppiamento stesso, canali con una potenza altamente variabile nel tempo possono essere disaccoppiati, mentre canali con bassa variazioni possono ricevere i parametri meno spesso.

2.4.2.1.4 Canale di Accoppiamento

Le coordinate di accoppiamento sono rese prendendo il rapporto di potenza da ciascuna banda dell'accoppiamento, la potenza nel canale originale è diviso per la potenza nel canale accoppiato, il rapporto di potenza diventa la coordinata. Le coordinate di accoppiamento sono convertite in formato floating point e quantizzate, questo genera le coordinate maestre di 2 bit per il canale (permettono al range dinamico rappresentato dall'accoppiamento di essere incrementato).

2.4.2.1.5 Rematixing

Se è attivo per ogni banda le misure di potenza sono fatte sia su L (canale sinistro) e R (canale destro) che su L+R e L-R, se la massima potenza si trova su

L+R o L-R il flag di rematrixing viene settato, ed in tal caso si codificano L+R e L-R invece di L e R. Questa tecnica è importante per preservare la compatibilità e mascherare il rumore di quantizzazione.
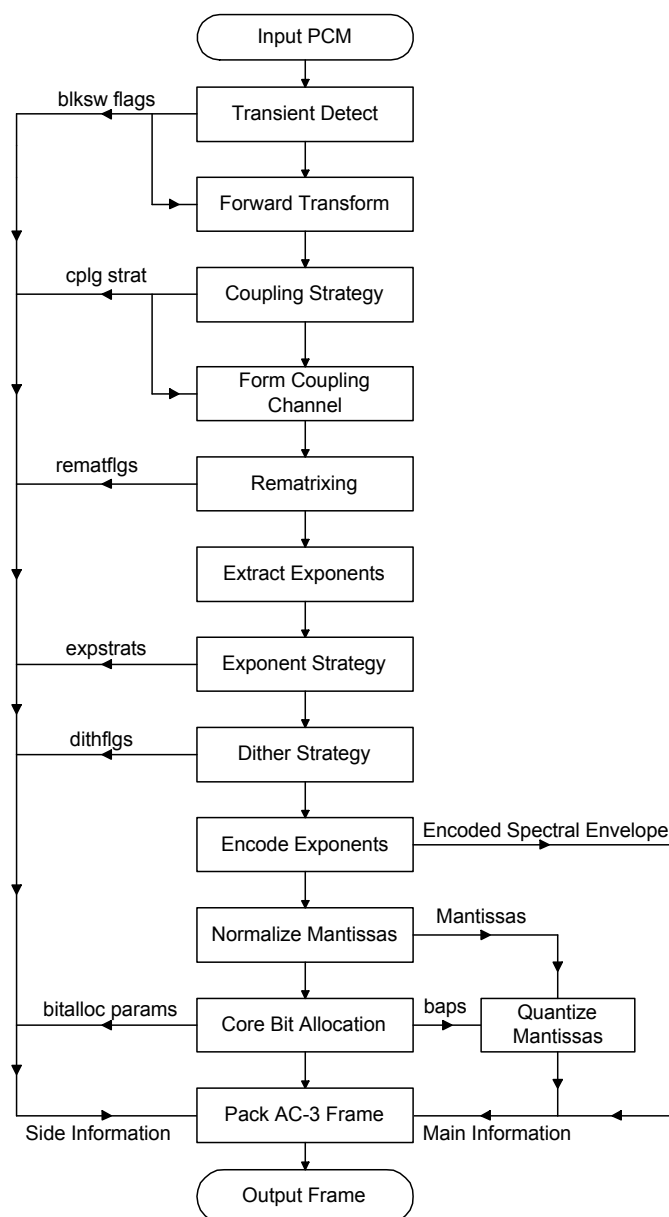
Immagine 2.22 - Flusso di Codifica AC3

2.4.2.1.6 Estrazione degli Esponenti

La rappresentazione binaria di ciascuna frequenza è esaminata per determinare il numero di zeri iniziali (*leading zeros*) , questo numero (fino ad un massimo di 24) diventa il valore esponenziale di base. Questi esponenti sono estratti e l'insieme utilizzato per determinare l'appropriata strategia.

2.4.2.1.7 Exponent Strategy

Per ciascun canale è esaminata la variazione degli esponenti rispetto alle frequenze, ed in base a ciò viene scelta la strategia adeguata. L'informazione riguardo la strategia è inclusa in ciascun blocco audio; l'informazione non è ma condivisa tra frame così il blocco 0 contiene la strategia ed i blocchi da 1 a 5 indicano il riutilizzo di esponenti precedenti all'interno dello stesso frame. Le diverse strategie permettono un tradeoff tra la richiesta di dati e la risoluzione della frequenza, in ciascuno dei modi il numero degli esponenti differenziali è combinato in parole di 7-bit, la differenza principale tra i vari metodi riguarda il modo in cui i coefficienti sono combinati insieme.

2.4.2.1.8 Dither Strategy

L'encoder controlla se i coefficienti che saranno quantizzati a zero saranno riprodotti con un dither (una forma di rumore con una opportuna distribuzione, che viene volontariamente aggiunto ai campioni con l'obiettivo di minimizzare la distorsione introdotta dal troncamento nel caso in cui si riquantizzino i campioni stessi). L'intento è di mantenere approssimativamente la stessa energia nello spettro riprodotto anche se non ci sono bit allocati in quella porzione dello spettro. A seconda della strategia degli esponenti e dell'accuratezza dell'encoder si può beneficiare dalla rimozione del dither per alcuni blocchi.

2.4.2.1.9 Codifica degli Esponenti

A seconda della strategia scelta gli esponenti sono preprocessati e codificati differenzialmente per la trasmissione nel bitstream; il preprocessing serve per assicurarsi che gli esponenti non assumino valori non legali (gli esponenti differenziali sono limitati nel range ±2), il risultato può portare alla diminuzione di alcuni coefficienti e la mantissa corrispondente avrà alcuni zeri iniziali. Gli esponneti sono codificati differenzialmente in modo da generare uno *spectral envelope.*

2.4.2.1.10 Normalizzazione delle Mantisse

I coefficienti trasformati di ciascun canale sono normalizzati tramite uno shifting sinistro pari al valore dell'esponente; i coefficienti della frequenza binaria originale sono shiftati in accordocon gli esponenti utilizzati dal decoder.

2.4.2.1.11 Allocazione dei Bit Centrali

In questa fase vengono allocati staticamente i bit centrali del pacchetto con valori fissati di default.

2.4.2.1.12 Quantizzazione delle Mantisse

Le mantisse normalizzate sono quantizzate dal quantizatore indicato dal bap (*bip allocation pointer*) corrispondente; le mantisse sono quantizzate arrotondandole al numero di bit indicato dal bap (Q. asimmetrica) o tramite una tabella (Q. simmetrica).

2.4.2.1.13 Creazione del Pacchetto AC3

I dati sono uniti in un pacchetto AC3, alcune delle mantisse quantizzate possono essere raggruppate insieme e codificate come uno stesso codeword.

Capitolo 3

Strumenti per la conversione e la distribuzione Video

La transocodifica (transcoding) consiste nella conversione digitale di contenuti multimediali, il termine è utilizzato in modi e definizioni diverse, in particolare:

- Conversione da un formato digitale all'altro, come dall'MPEG-2 all'MPEG-4, questa è la definizione più corretta.
- Modifica del bitrate dello stream per adattarsi alle specifiche richieste della rete o del dispositivo, questo processo è chiamato anche *transrating*.
- Cambiamento della risoluzione, tipicamente degradamento dalla qualità HD ad una risoluzione standard o inferiore.

Il processo di transcodifica seguito parte dalla presenza di un certo numero di file video in formato .vob estratti dalla DVD-Video ed individua tre passaggi principali per la loro trasformazione.

1. Separazione dei contenuti audio e video (chiamato anche demultiplexing) ed indicizzazione della parte video come passo di preprocessing.
2. Conversione della traccia audio tramite decoding dal formato di partenza e successiva ricodifica nel formato desiderato.
3. Applicazione delle tecniche di editing video ed unione della traccia audio con quella video.

In questo capitolo esploreremo gli strumenti di cui ci serviamo per effettuare la conversione partendo da qualche accenno sulla struttura di base di un DVD, per poi specificare i vari software sfruttati per ciascuno di questi passi. Nell'ultima parte del capitolo ci concentreremo sullo streaming di contenuti multimediali, parlando inizialmente della teoria di base ed in seguito dell'utilizzo pratico.

3.1 Struttura di un DVD Video

Un DVD video è composto da una sottodirectory VIDEO-TS che contiene tutti i dati riguardanti il filmato; di seguito mostriamo un esempio di directory VIDEO_TS all'interno di un DVD:

22.08.2000 05:23 12'288 VIDEO_TS.BUP
22.08.2000 05:23 12'288 VIDEO_TS.IFO
22.08.2000 05:23 333'824 VIDEO_TS.VOB
22.08.2000 05:23 59'392 VTS_01.BUP
22.08.2000 05:23 59'392 VTS_01.IFO
22.08.2000 05:23 8'192 VTS_01.VOB
22.08.2000 05:27 1'073'643'520 VTS_01.VOB
22.08.2000 05:32 88'064 VTS_02_0.BUP
22.08.2000 05:32 88'064 VTS_02_0.IFO
22.08.2000 05:32 59'379'712 VTS_02_0.VOB
22.08.2000 05:36 1'073'436'672 VTS_02_1.VOB
22.08.2000 05:40 1'073'549'312 VTS_02_2.VOB
22.08.2000 05:45 1'073'502'208 VTS_02_3.VOB
22.08.2000 05:49 1'073'371'136 VTS_02_4.VOB
22.08.2000 05:53 1'073'555'456 VTS_02_5.VOB
22.08.2000 05:57 810'952'704 VTS_02_6.VOB

Come si nota ci sono tre diversi tipi di files: .VOB, .IFO e .BUP, essi sono suddivisi in gruppi formati da un file IFO, un file BUP ed almeno un file VOB, ciascun gruppo rappresenta un contenuto multimediale ben definito, ed è riconoscibile dal fatto che tutti i file condividono lo stesso nome cambiando unicamente l'estensione. Ciascun DVD contiene sempre un gruppo VIDEO_TS che contiene le informazioni di visualizzazione ed eventualmente il menù; in seguito sono indicati i gruppi nominati come VTS_XX che rappresentano i filmati veri e propri. In un DVD video possono essere contenuti fino a 99 filmati differenti, e ciascuno di essi può essere memorizzato in un massimo di dieci file VOB (che ha dimensioni massime di 1GB) nominati da VTS_XX_0 a VTS_XX_9; in questo caso i file IFO e BUP fanno riferimento

3.1.1 File VOB

Un file VOB (Video OBjects) contiene alcuni flussi di informazioni multimediali contemporanei (multiplexati): video, audio e sottotitoli; un file VOB all'interno di un DVD, specialmente se ad uso commerciale, è tipicamente criptato, la fase di decrittazione e memorizzazione dei file VOB è tipicamente il primo passo in una conversione ed è chiamata fase di *ripping*. Il video è in formato MPEG-2, l'audio può essere di tipo AC-3, Linear PCM, Mpeg-2 multicanale oppure MPEG1 layer2 a 2 canali audio. L'AC3 è quello standard mentre l'MPEG-2 multicanale si può trovare solo su pochissimi dischi in quanto tale formato fu inizialmente proposto come standard per la Regione 2 (Europa e Giappone) e poi fu abbandonato. Il PCM è principalmente utilizzato sui DVD contenenti musica mentre l'MP2 è solo nelle

produzioni ultra economiche. Il PCM è il tipo audio high quality non compresso e richiede molto spazio, quindi non è una scelta molto pratica per un intero film, spesso con possibilità di audio su più lingue, scena extra, interviste, ecc. L'audio in formato AC3 ha un bitrate compreso fra 192 e 448 kbit/s (192 kbit/s per un audio a due canali stereo, da 384 a 448 kbit/s per l'audio 5.1 surround). Un file VOB può contenere il flusso video principale (main stream) e alcuni multiangle streams, con i quali si può cambiare (ad esempio) il punto di vista durante il film; questa caratteristica viene utilizzata principalmente per mostrare gli storyboards oppure per interventi extra durante la visione del film. Il massimo bitrate del flusso video è di 9.8 Mbit/s, sommati assieme, video ed audio non devono superare i 10 Mbit/s. Si possono avere fino a 9 canali audio (in lingue diverse) liberamente scambiabili fra loro durante la visione del film e fino ad un massimo di 32 diverse sequenze di sottotitoli costituiti da bitmap a 4 color sovrapposti (in overlay) alla normale sequenza video e codificati separatamente rispetto al video. Di seguito mostriamo un esempio del contenuto di un file VOB estratto da un programma di ripping.

```
Found 0xBF = Private 2 [@LBA 0]
Found VOB-ID: 01/CELL-ID: 01 [@LBA 0]
Encountered encrypted sector, attempting key recovery [@LBA 1]
Deduced key: 0xC00374C61C (2/2 vkey(s))
Found 0xE0 = Video 0 [PTS 0:00:00.290 @LBA 1]
Width = 720
Height = 480
Aspect-ratio = [3] 16:9 display
Frame-rate = [4] 29.97 (30000/1001) fps
Found 0xBD = Private 1, sub 0x80 [PTS 0:00:00.224 @LBA 99]
Found 0xBD = Private 1, sub 0x81 [PTS 0:00:00.224 @LBA 100]
Found 0xBD = Private 1, sub 0x82 [PTS 0:00:00.224 @LBA 101]
Found 0xBD = Private 1, sub 0x83 [PTS 0:00:00.224 @LBA 102]
Found 0xBE = Padding [@LBA 117]
Found 0xBD = Private 1, sub 0x20 [PTS 0:00:00.724 @LBA 169]
Found 0xBD = Private 1, sub 0x21 [PTS 0:00:00.724 @LBA 170]
Found VOB-ID: 02/CELL-ID: 01 [@LBA 378]
```

Come già detto in precedenza, normalmente c'è un solo video ed ha sempre l'ID 0xE0. Il PTS è il Program Time Stamp e ci dice quando inizia il video rispetto all'inizio del file VOB. Si può anche vedere che la risoluzione è di 720x480 quindi si tratta di un DVD in formato NTSC (americano). Il numero di fotogrammi al secondo (frame rate) pari a 29.97 conferma questa deduzione. Il DAR (display aspect ratio = rapporto di aspetto dell'immagine N.d.T.) è 16:9. A seguire, si vedono quattro canali audio agli ID fra 0x80 e 0x83; questi canali sono tutti di tipo AC3 poichè hanno il codice 0xBD, i cui sottocanali sono i già citati 0x8x. Nel caso l'audio sia codificato in DTS esso avrà ID 0xBD 0x88 oppure 0xBD 0x89; se fosse PCM avrà

ID 0xBD, con i canali aventi ID 0xAx, da 0xA0 fino a 0xA9. Infine, un eventuale audio MP2 avrebbe indirizzo 0xCx. Ecco un esempio di audio PCM ed MP2:

```
Found 0xBD = Private 1, sub 0xA0 [PTS 0:11:01.479 @LBA 4]
Found 0xC0 = Audio 0 [PTS 0:00:00.440 @LBA 25]
```

Gli ultimi due stream 0x20 e 0x21 sono i sottotitoli, anch'essi appartenenti al flusso avente l'ID 0xBD. Inoltre ci sono di solito dei "padding streams" (0xBE) utilizzati per preservare il bitrate. Infine l'informazione VOB-ID e Cell-ID. Dal punto di vista del bitstream un file VOB è codificato esattamente come un file .mpeg, ed è formato da streams di 2048 bytes autocontenuti e può essere letto da un qualsiasi player capace di leggere i file mpeg (ad esempio Quicktime).

3.1.2 File IFO e BUP

I files tipo IFO (InFOrmation) forniscono al player informazioni importanti per lo svolgimento, quali il punto di inizio di un capitolo, dove si trovano un certo canale audio oppure un sottotitolo in lingua e così via, questo è il motivo per cui è possibile 'rippare' alcune parti di un DVD (es. un capitolo) mediante un software che possa leggere queste informazioni.

```
Parsing "f:\video_ts\vts_02_0.ifo"...
0. Length: 02:07:15:24 in 15 cell(s).
1. Length: 00:00:01:01 in 1 cell(s).
```

In questo caso vengono mostrati 2 PGC (ProGram Chains) per il film in esame. Il PGC 0 rappresenta il film principale mentre il PGC 1 contiene i logo della casa produttrice mostrati all'inizio della sequenza. La struttura dei PGC può anche essere più complessa, anche se normalmente è semplice come nell'esempio indicato. I film con multiangle conterranno più PGC tutti della stessa lunghezza, mentre i seamless branching movies avranno più PGC di lunghezze diverse. Selezionando il PGC giusto, si ottiene la corrispondente versione del film. Ed ancora... il PGC corrisponde al numero della scena mostrata dal player. Infine, occorre notare che i files IFO non sono criptati. Infine i files tipo BUP (BackUP) sono solamente dei backup dei corrispondenti IFO e, come questi, sono in chiaro.

3.1.3 Struttura di un DVD

Una Cella è la più piccola unità di riproduzione presente su di un DVD, può essere ad angolo singolo o multiangolo, contenere o meno uno o più flussi audio, ed è identificata univocamente da un insieme di cifre il Cell-ID (da 0 a 255) ed eventualmente da VOB-ID (da 0 a 65535) che rappresentano le varie unità elementari all’interno della cella (ad esempio i vari angoli).

Immagine 3.1- Struttura della Celle di un DVD

Una sequenza di celle con numeri di identificazione successivi è definita come un programma, usato principalmente per la riproduzione casuale o ripetuta oppure come riferimento ad un gruppo di celle nella ricerca dei capitoli. Più programmi di nuovo con identificatori in sequenza è definito come Parte di un Titolo (*Part-Of-Title o PTT*) e tipicamente indicano i capitoli del filmato e sono rappresentati come le unità di accesso casuale a cui fa riferimento l'utente. Un insieme di celle o di programmi sono tipicamente assegnati ad una *Program Chain* (PGC), che definisce l'ordine ed il modo in cui le celle sono riprodotte, a seconda delle impostazioni del player DVD.

Immagine 3.2- Struttura del Capitolo di un DVD

Le specifiche del DVD non richiedono necessariamente ai capitoli o PTT di appartenere allo stesso PGC, l'immagine sottostante mostra quattro capitoli suddivisi tra due PGC distinti; sebbene l'identificatore dei capitoli aumenti sequenzialmente (nell'immagine il numero del capitolo è indicato dall'etichetta accanto all'icona del

programma) il numero del programma riparte da 1 per ogni PGC. Differenti PGC possono puntare alla stessa cella (ad esempio i due PGC dell'immagine hanno le celle).
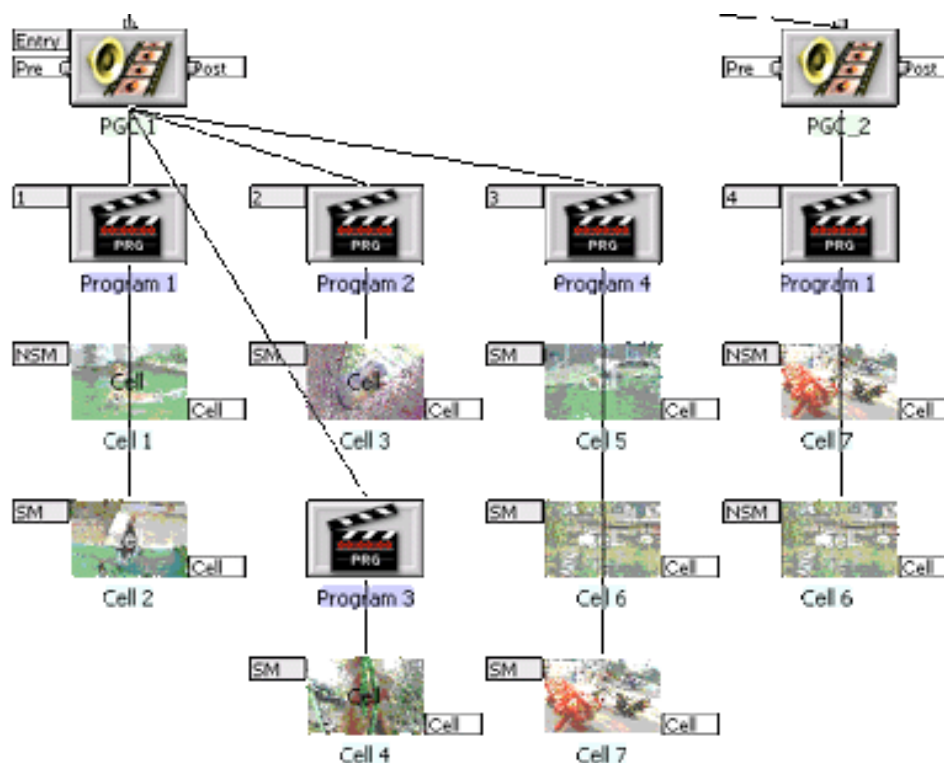
Immagine 3.3- Struttura generale di un DVD

3.2 Indicizzazione e Demultiplexing

Questa fase consiste nel processare i file VOB in cui è memorizzato il filmato da convertire, memorizzati nell'Hard Disc del computer dopo un eventuale processo di ripping nel caso essi fossero criptati, attraverso il software DGIndex. Tale passaggio restituisce un file in formato D2V, un file temporaneo (*pass-through file*) il cui compito è indicare alle applicazioni di editing l'ordine ed il modo in cui processare i file video a cui è riferito. Un altro compito che esegue quest'applicazione consiste nel effettuare il demultiplexing tra il flusso video e l'audio, dandoci la possibilità di processarli separatamente. Di seguito descriveremo il funzionamento del software ed alcune delle operazioni che è capace di compiere.

3.2.1 DGIndex

DGIndex, parte del package DGMPGDec, è un piccolo ma potentissimo tool il cui scopo primario è quello di decodificare streams MPEG-1 e 2 e creare un indice contenente, per ogni frame, le locazioni ed altre informazioni, prelevandole dagli header; questo indice, o file progetto, è utilizzato in seguito da altri processi suc-

cessivi per accedere e processare i file video. DGIndex è in grado di decodificare la maggior parte degli stream MPEG 1 e 2, inclusi streams elementari (PES), program-streams, transport-streams, e file che sfruttano questa codifica come VOB, VCD (video CD), SVCD (super video CD), ed altri; addizionali includono video demuxing (nei formati m1v e m2v), audio demuxing, audio demuxing (nei formati ac3, dts, aac, mpa, lpcm e wav), iDTC ottimizate, filtri di luminosità e cropping (taglio di alcune parti dell'immagine video). Di seguito indichiamo attraverso i menù della GUI le principali opzioni del software che interessano nella decodifica del video.

- ***Video:*** questo menù contiene le opzioni da impostare per la docodifica ed alcuni filtri.
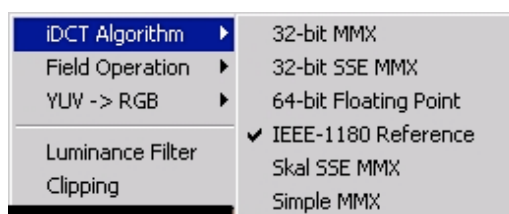
Immagine 3.4- Menù video di DGIndex

  - *iDCT Algorithm*- seleziona l'algoritmo di iDCT (inverse discrete cosine transform), se nel caso della DTC l'algoritmo è uno solo, nel caso dell'iDTC gli algoritmi sono svariati (e ognuno ha un suo margine di errore); lo standard IEEE-1180 definisce un margine di errore entro il quale l'iDTC in uso da risultati "accettabili", quindi utilizzando un qualsiasi iDTC che rispetti lo standard siamo sicuri di ottenere un risultato qualitativamente buono, al contrario avremo errori di decodifica. In DGIndex sono disponibili 7 algoritmi diversi:
  - *Field Operation*- Field Operation, permette di decidere come gestire il flag RFF (REPEAT_FIRST_FIELD) che, nello stream MPEG-2, serve per replicare il frame che contiene tale flag al fine di realizzare il 3:2 pulldown, utilizzato per permettere ai sistemi NTSC (29,97 fps) la visualizzazione di video a 23,976 fps.
  - *YUV -> RGB*- effettua una conversione del colorspace su due scale differenti:
    - YUV [16, 235(Y)/240(UV)] => RGB [0, 255] (PC scale)
    - YUV [16, 235(Y)/240(UV)] => RGB [16, 255] (TV scale)
  - *Luminance Filter e Clipping*- permettono di regolare la luminosità, un pò come avviene con alcuni filtri per avisynth, e di croppare l'immagine.
- ***Audio:*** contiene le opzioni per il demultiplex delle tracce audio e l'editing.
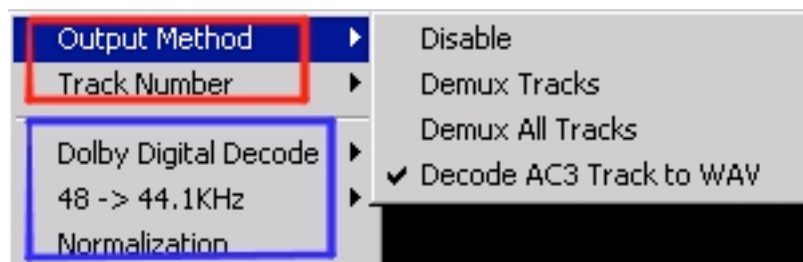
Immagine 3.5- Menù audio di DGIndex

- *Output Method e Track Number-* specificano cosa estrarre con esattezza, a seconda della selezione, si abiliteranno o meno le altri parti del menù Audio.
  - *Demux All tracks:* estrae tutte le tracce così come sono. In questo caso, sia Track Number che le altre voci del menu Audio saranno disabilitate.
  - *Demux tracks:* permette di scegliere quali tracce estrarre attraverso Track Number, che stavolta risulterà abilitato
  - *Decode AC3 Track to WAV:* abilita la 2° parte del menù che permette di effettuare delle normalizzazioni o delle conversioni del sample rate oltre che a convertire direttamente le tracce AC3 estratte in WAV.

DGIndex oltre che a fornire il file .d2v è molto utile in fase di analisi dello stream, in particolare per quello MPEG-2; tale analisi si può effettuare in due modi differenti che non sono mutuamente esclusivi, anzi spesso complementari. Tali sistemi sono:

1. *Visualizzazione dell'anteprima:* L'anteprima è composta da due finestre, una in cui viene riprodotto il video ed riquadro *Information* che contiene alcuni valori significativi ed è composta da tre sezioni: Video, Audio e Status:

- ***Aspect Ratio:*** Mostra semplicemente l'AR ad es. 4:3, 16:9 o 2.21:1.
- ***Frame Size:*** E' la risoluzione. Nei DVD commerciali contenenti un video MPEG-2 il valore tipico è 720 x 576 (o 720 x 480).
- ***Profile:*** Corrisponde al modi encodare i dati del video. Se abbiamo un video MPEG, il valore di Profile sarà MPEG1 (vedi fig.2 e nota anche come può cambiare la risoluzione).
- ***Frame Rate:*** E' il framerate del video.
- ***Video Type:*** Se non ci sono i flags del pulldown e il framerate corrisponde a 25, Video Type assumerà il valore PAL, altrimenti NTSC.
- ***Frame Type:*** assume come valori Interlaced o Progressive. Il valore di questo campo non stabilisce con certezza che un video sia interlaced o progressive ma è la lettura del flag PROGRESSIVE_FRAME del frame che indica in quale modo

il medesimo sia stato encodato; se un video è veramente interlaced, il valore di Frame Type sarà interlaced come ci aspettiamo, il viceversa non vale.
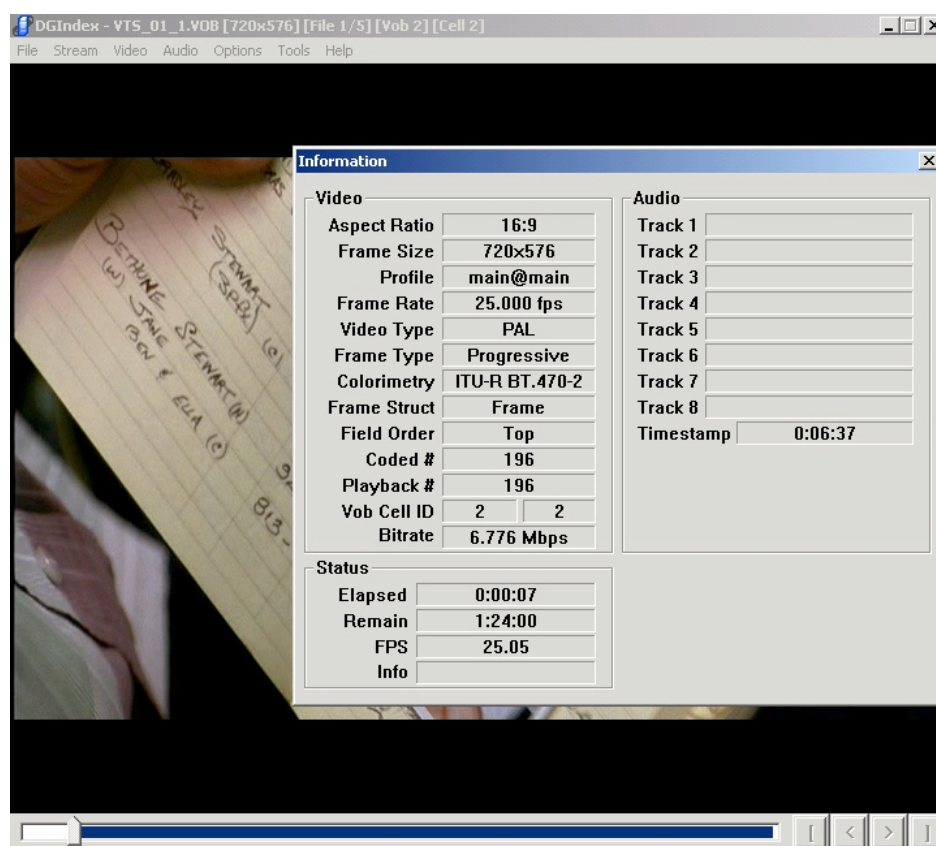
Immagine 3.6- Analisi video in DGIndex

1.1. Video

- ***Colorimetry:*** Mostra lo schema dei coefficienti di colori usato. La conoscenza di tale schema è molto utile in fase di codifica perchè è possibile rettificare il modo di rappresentare i colori, onde evitare il leggero sbiadimento tipico dei codec MPEG-4; indica se sia necessario o meno rettificare la gestione dello schema dei colori in fase di codifica.
- ***Frame Struct:*** Un frame può essere strutturato per frame o per field quando un frame contiene l'immagine intera o una parte di essa (i campi)
- ***Field Order:*** Indica l'ordine dei visualizzazione dei campi. Può valere top o bottom. E' pertinente quando Frame Struct vale field perchè indica in quale ordine visualizzare i campi. Se il frame contiene l'immagine intera, Field Order perde di significato.
- ***Coded #:*** E' il numero di immagini decodificate da DGIndex durante l'anteprima o il salvataggio del progetto.

- ***Playback #:*** E' il numero di immagini mostrate da DGIndex durante l'anteprima o il salvataggio del progetto.
- ***Vob Cell ID:*** Valido solo per gli MPEG-2 su VOB, questo valore mostra la posizione del frame corrente all'interno della matrice (VOB, CELL)
- ***Bitrate:*** Calcola il bitrate medio ogni 64 frame

1.2. Audio

- ***Track [i]:*** Mostra il tipo di audio e il suo bitrate
- ***Timestamp:*** Mostra il timestamp per la posizione corrente dello stream audio

1.3. Status

- ***Elapsed:*** Mostra il tempo trascorso dall'inizio del salvataggio del progetto o dall'avvio dell'anteprima.
- ***Remain:*** Mostra il tempo rimanente al completamento del salvataggio del progetto o alla conclusione dell'anteprima.
- ***FPS:*** E' il framerate medio relativamente però alla velocità con cui DGIndex processa i frame, se la velocità di playback per esempio viene variata attraverso l'apposito menù, tale valore rifletterà questa variazione.
- ***Info:*** Info di Debug o eventuali informazioni di errore.

2. *Esame dell'output prodotto da parseD2V:* Parse D2V effettua una scansione del file .d2v e restituisce un file di testo avente lo stesso nome del file d2v e il cui contenuto è la versione compattata di alcune importanti informazioni presenti nel file d'origine, senza che ci sia bisogno di esaminare i flags bit per bit. Ecco un esempio del file ottenuto:

```
Encoded Frame: Display Frames....Flags Byte (* means in 3:2 pattern)
--------------------------------------------------------------------
[Field Operation None, using flags]
[GOP]
[Clip is TFF]
0 [I]: 0,0......................2
1 [B]: 1,1......................2
2 [P]: 2,2......................2
3 [B]: 3,3......................2
4 [B]: 4,4......................2
5 [P]: 5,5......................2
6 [B]: 6,6......................2
7 [B]: 7,7......................2
8 [P]: 8,8......................2
9 [B]: 9,9......................2
10 [B]: 10,10...................2
11 [P]: 11,11...................2
[GOP]
12 [B]: 12,12...................2
13 [B]: 13,13...................2
14 [I]: 14,14...................2
15 [B]: 15,15...................2
16 [B]: 16,16...................2
...
```

Le informazioni sui flags contenuti nel file d2v sono immediatamente disponibili. Il file riporta nell'ordine:

- l'ordine dei frame (TFF o BFF)
- L'inizio e la fine del GOP
- l'indice del frame
- il tipo (I, P, B)
- la coppia di indici dei frames presenti nei 2 fields
- il pattern rappresentato dalla coppia di flags (TFF, RFF). Il numero è decimale e può assumere come valore 0,1,2,3 che rappresentano tutte le possibili combinazioni binarie dei flags.

In questo file è subito evidente come:

- non vi sia nessun pattern particolare (è una sequenza costante di 2), o meglio, vi è il 2:2 che non provoca nessuna ripetizione di campi.
- i fields contengano l'intera picture, quindi non vi è nessun interlacciamento.

DGIndex può essere eseguito anche tramite linea di comando, questa caratteristica permette l'uso del software in modo automatico tramite script, processi batch o l'uso di applicazioni di terze parti.

3.2.2 Esame del file d2v

Un esempio di file d2v restituito da DGIndex è il seguente:

```
CODE
DGIndexProjectFile11
3
J:\FILM\VIDEO_TS\VTS_01_1.VOB
J:\FILM\VIDEO_TS\VTS_01_2.VOB
J:\FILM\VIDEO_TS\VTS_01_3.VOB
Stream_Type=1
MPEG_Type=2
iDCT_Algorithm=5
YUVRGB_Scale=1
Luminance_Filter=0,0
Clipping=0,0,0,0
Aspect_Ratio=16:9
Picture_Size=720x576
Field_Operation=0
Frame_Rate=25000
Location=0,0,4,2F0EE
d00 1 0 2048 1 1 92 f2 e2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 184320 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 372736 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 593920 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 819200 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 1101824 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 1290240 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 1515520 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
900 1 0 1730560 1 1 72 72 d2 f2 f2 e2 f2 f2 e2 f2 f2 e2
...
```

Il numero in alto corrisponde al numero dei VOB, se lo stream è MPEG-1 sarà sempre 1; successivamente vengono riassunte sia le impostazioni dell'utente che alcune caratteristiche dello stream (Stream_Type, MPEG_Type, Aspect_Ratio, Picture_Size, Frame_Rate, Location), infine, c'è una serie di cifre, alcuni di questi valori sono esadecimali, da convertire in binario perchè ogni bit corrisponde all'attivazione di un flag. I primi 5 valori sono:

info(Hex): insieme di informazioni relative ai GOP. Sono 11 bit di cui i primi 8 da destra sono riservati.

matrix: altri non è che il ColorMatrix

file: è l'indice del VOB, per n VOB esso ha range [0, n-1].

position-vts-cell: è la terna che indica la posizione dei frames da decodificare nel DVD.

Questo è il vero indice che DGindex costruisce per l'accesso ai frames, tutto il resto sono informazioni prelevate dagli header che descrivono la natura del frame.

Consideriamo la prima riga dell'esempio, info corrisponde a d00, per cui avrei in binario: 110100000000, i primi 8 bit significativi sono riservati, gli ultimi quattro valgono da destra verso sinistra:

8° bit: vale 1 se l'immagine I apre un nuovo GOP.

9° bit: vale 1 se le immagini della linea fanno parte di una sequenza progressiva. 0 altrimenti

10° bit: vale 1 se le immagini della linea fanno parte di un GOP chiuso. 0 altrimenti

11° bit: vale sempre 1 per indicare l'inizio di una nuova linea dati

I rimanenti valori sono relativi ai frames, decodificati, uno per uno, sempre in relazione all'esempio di prima, prendiamo in esame l'ultimo numero della prima riga, e2 che in binario diventa: (11)(10)(00)(10). Il 3° e 4° bit sono riservati, gli altri da destra verso sinistra:

1° e 2° bit: rispettivamente REPEAT_FIRST_FIELD e TOP_FIELD_FIRST (RFF e TFF flags), sono i flags che si utilizzano per capire se c'è un 3:2 pulldown

5° e 6° bit: PICTURE_CODING_TYPE flags, indica se un frame è di natura I, P, B.

- 00 (Riservati), 01 (I-Frame), 10 (P-Frame), 11 (B-Frame)

7° bit: PROGRESSIVE_FRAME flag, vale se il frame è codificato come interlaced, 1 come progressive.

8° bit: vale 1 se il frame è decodificabile in modo indipendente dal GOP precedente, 0 altrimenti.

3.3 Conversione Audio

In questa fase si effettua la transcodifica dei flussi audio dal formato originale AC3 al formato che si vuole utilizzare (nel nostro caso l'MP3) per far ciò si utilizzano due software diversi, il primo è un AC3 decoder (Azid), il secondo un MP3 encoder (lame), tra il primo ed il secondo passo i dati vengono mantenuti da un file intermedio in formato WAV. Un file WAV (o WAVE), contrazione di WAVEform audio format (formato audio per la forma d'onda) è un formato audio sviluppato da Microsoft e IBM per personal computer IBM compatibile supporta varie modalità di immagazzinamento dei dati ma nella pratica il più diffuso è il metodo PCM che provvede a salvare i dati audio senza nessun tipo di compressione, la forma d'onda viene memorizzata direttamente, quindi i file risultanti sono di elevate dimensioni, ma non richiedono elevata potenza di calcolo per essere riprodotti.

3.3.1 Azid

Azid è un programma a linea di comando il cui compito è decodificare uno stream in formato audio AC3 in un formato intermedio; il processo di decodifica segue le specifiche dello standard dell'ASTC. Una delle feature più importanti di questo software consiste nelle varie opzioni per la gestione dei canali del flusso AC3, che può trasportare fino a 6 canali contemporaneamente. Come primo passo il decoder processa un blocco audio separando il flusso nei suoi canali elementari, essi rappresentano gli stessi canali acquisiti dal codificatore AC3 al momento della creazione (destro, sinistro, centrale, ecc...); se il numero dei canali di output è minore del numero dei canali codificati sarà necessario eseguire un downmix nel numero corretto. Questi canali sono l'imput del processo di decodifica e sono chiamati *left* (l), *right* (r), *center* (c), *surround left* (sl or s), *surround right* (sr) e *low-frequency effect channel* (lfe); l'operazione di downmix riduce il numero di canali, ed è controllata dall'opzione -d della linea di comando, che selezione il numero di canali principali e “rear”. Una volta eseguita questa operazione l'audio così ottenuto è dato in ingresso al selettore dell'output che specifica per ciascun ingresso l'effettivo canale di output a cui è assegnato ed è controllato dall'opzione -o. Inoltre Azid permette di specificare separatamente per ogni canale gli attributi, che consistono nel guadagno assegnato al canale o nel metodo di compressione dinamica

specificata per quel canale. Azid specifica le seguenti opzioni per la compressione dinamica che consiste nell'amplificare o attenuare il suono per diminuire la variazione dinamica globale (*loudness*):

- *No dynamic compression:* Non utilizza la decompressione dinamica.
- *Normal dynamic compression:* Metodo di decompressione dinamica di default.
- *Light dynamic compression:* Questa opzione ha il 50% del rapporto tra guadagno/riduzione rispetto al metono di decompressione normale.
- *Heavy dynamic compression:* Inserito per migliorare la qualità in caso di suoni ambientali con molto rumore di fondo.
- *Inverse Dynamic expansion:* Consiste nell'inverso rispetto alla compressione dinamica leggera, amplifica i rumori forti e attenua quelli deboli.

3.3.2 LAME

LAME è un software a linea di comando utilizzato per le codifiche audio in formato MP3, il nome è un acronimo ricorsivo per *LAME Ain't an MP3 Encoder* (LAME non è un codificatore MP3) e riflette le sue prime implementazione quando il software non era un codificatore. Nel 2004 era opinione comune tra la comunità degli audiofili secondo cui LAME produceva audio MP3 con la maggiore qualità nel caso di bitrate di 128 kbps o superiori. LAME è abile di effettuare codifiche utilizzando tre diversi metodi:

- ***Constant Bitrate (CBR):*** Questo è il metodi di default ed il più basilare, in questo caso il bitrate è mantenuto costante per tutto il file, ciò significa che la qualità della compressione audio è variabile a seconda della facilità o difficoltà di compressione dei vari pezzi del flusso audio originale, le parti semplici avranno una qualità decisamente maggiore rispetto alle parti complesse dato che hanno a disposizione la stessa quantità di dati ma richiedono meno informazione. Il vantaggio principale consiste nella facile ed accurata prevedibilità delle dimensioni dei file di output.
- ***Average Bitrate (ABR):*** In questo metodo si specifica un bitrate ed il decodificatore cercherà di mantenere la media del bitrate tra le varie parti del file il più costante possibile, migliorando nel contempo la qualità del risultato dato che si utilizzano bitrate maggiori nelle parti più complesse. Questo metodo è maggiormente raccomandato rispetto al CBR perché pur migliorando la qualità permette ancora una facile predizione delle dimensioni dell'output.

- ***Variable bitrate (VBR):*** Questo metodo permette di scegliere la qualità desiderata in una scala tra 9 (qualità minore/distorsione maggiore) e 0 (qualità maggiore/distorsione minore) ed il codificatore tenta di mantenere la medesima qualità per tutto il file scegliendo il numero ottimale di bit da utilizzare per ciascuna parte del flusso. Questo metodo produce file in output con la migliore qualità e permette di specificarne il livello, l'inconveniente è che la dimensione del file risultante è totalmente imprevedibile.

3.4 Conversione Video

Questa fase consiste nella transcodifica degli stream video, accompagnata da eventuali operazioni di editing; gli strumenti di cui ci serviamo in questa fase sono il software Avisynt ed il programma VirtualdubMod.

3.4.1 Avisynt

Avisynt agisce come un editor di video non-lineare, privo di una GUI e controllato interamente da script, si pone come un intermediario tra la sorgente video digitale (tipicamente un file .avi o un video mpeg) ed un programma ricevente, tipicamente un media-player, un software di editing video o un encoder. L'editing è non-lineare quando gli interventi sui frame non sono condizionati dalla loro sequenzialità, i frames possono essere ritoccati, ricostruiti, rimontati in un ordine che non necessariamente è quello iniziale; un esempio di editing lineare è invece il montaggio su una VHS. Avisynt comunica con ogni tipo di programma che supporta i file .avi agendo come una falsa sorgente di dati, permettendo così l'elaborazione senza ricorrere ad un file intermedio; inoltre può applicare una gran varietà di funzioni di editing e di filtri ad uno stream video, prima che esso venga effettivamente letto dal programma.questa tecnica è chiamata *frameserving*, il frameserving di avisynth offre molti vantaggi:

- La possibilità di intervenire potenzialmente su ogni singolo frame del video
- La possibilità di disporre di un numero sterminato di filtri
- L'aumento delle prestazioni nella codifica.

La capacità dei filtri include il trimming (selezione di una sequenza ristretta di frame video), cropping (taglio di alcune parti dell'immagine video), deintelacing (eliminazione dell'interlacciamento), manipolazione di immagini singole, correzione dei colori, eliminazione del rumore e molto altro; tecnicamente agisce come un compilatore per il proprio linguaggio si scripting, che agisce come un linguaggio di programmazione vero e proprio ed include numerose feature, utilizzo di variabili,

tipi di dati distinti, operatori aritmetici e condizionali espressioni complesse, ecc..., inoltre può essere esteso tramite l'uso di plugin esterni. I filtri di Avisynt agiscono in diversi spazi del colore, inclusi RGB, ; questo è necessario per permettere la compatibilità con ogni tipo di audio e video in input e output, alcuni filtri lavorano solo su spazi del colore specifici rendendo necessaria una conversione. Il linguaggio tratta principalmente video come dati primitivi, un tipico script carica in input un video e vi applica un processo, il video modificato è l'output dello script che viene interpretato dal programma che esegue lo script come il video originale. La maggior parte di funzioni Avisynth lavora su un clip ossia una sequenza di frame, il risultato viene acquisito implicitamente oppure può essere memorizzato in una variabile se si vogliono eseguire delle elaborazioni più fini; l'applicazione delle funzioni sui clip può avvenire:

- invocando direttamente la funzione.
- utilizzando una variabile e applicandovi la funzione per mezzo dell'operatore "

Un esempio di script è il seguente:

```
#Il DGDecode serve per accedere all'indice dei frames
LoadPlugin("C:\PROGRAMMI\DGMPGDec\DGDecode.dll")
#SORGENTE
mpeg2source("c:\film.d2v", idct=0)
#CROP, per il taglio delle bande nere
crop(6,14,708,548)
#RESIZE
BilinearResize(512,272)
```

Esso specifica le quattro operazione essenziali nell'economia della codifica MPEG-4, esse consistono in:

- ***Specificare la sorgente:*** L'acquisizione dalla sorgente avviene attraverso un file particolare di tipo d2v prodotto da DgIndex. La sorgente non è necessariamente un file di questo genere. Avisynth può processare benissimo tante altre tipologie di file caricandoli con le istruzioni opportune.
- ***Crop:*** Le misure per il crop possono essere prese usando GordianKnot (vedi guida) a mio avviso, il modo più semplice e preciso per croppare rispettando l'aspect-ratio (le proporzioni rispetto al video orginale)
- ***Resize:*** La scelta del resize può dipendere da tanti fattori, in funzione della precisione con cui il resize viene effettuato, si può tracciare la seguente gerarchia: Bilinear -> Bicubic -> Lanczos, tenendo presente che tanto più l'immagine risulta dettagliata, quanto meno risulta comprimibile con il rischio, se non si dispone di

un bitrate sufficiente, di generare artefatti inattesi (blocking). Dal punto di vista della qualità, si tenga presente questa regola più o meno generale: Se l'immagine viene rimpicciolita è meglio usare dei filtri poco aggressivi (bilinear o soft bicubic), al contrario se l'immagine viene ingrandita è bene usare un filtro più aggressivo (sharp bicubic o Lanczos). Nel primo caso si evita che l'immagine diventi inutilmente spigolosa senza alcun reale beneficio, nel secondo caso, un filtro poco aggressivo rende l'immagine ingrandita un pò sfumata e quindi poco chiara.

L'ordine di tali operazioni non è casuale: crop -> filtri -> resize. Normalmente il resize avviene dopo l'applicazione dei filtri, i quali sono successivi al crop; soprattutto per ragioni di pulizia del video, ma anche perché un denoise (i filtri che attenuano il rumore video) è più efficiente se applicato sui frame originali, in modo da eliminare il più possibile il rumore senza toccare i dettagli rilevanti. Il resize apporta delle approssimazioni e se il denoise lavorasse sul risultato del resize lo farebbe in modo scorretto, anche se in modo più rapido. Ora analizzeremo con cura lo script, perchè le operazioni evidenziate sono quelle che ritroverete nel 99% delle codifiche elementari (da usare quindi come base per script futuri)

Se si vuole migliorare la comprimibilità si può ricorrere all'uso di un paio di semplici tecniche:

1. Riduzione della luminosità
2. Riduzione minimale del dettaglio

Il senso della prima nasce dal fatto che i codec MPEG-4 hanno una difficoltà congenita nel comprimere le scene chiare a differenza di quelle scure che sono ottimamente comprimibili, riducendo un pochino la luminosità si aumenta la comprimibilità senza che le immagini siano troppo degradate, per il dettaglio c'è un discorso molto simile, riducendolo in minima parte, non sarà comunque visibile ad un'analisi superficiale del video, a tutto vantaggio della comprimibilità.

3.4.2 Plugin esterni

Oltre alle funzionalità di base di avisynt nel progetto utilizziamo alcuni plugin esterni:

DGDecode: DGDecode è parte del package DGMPGDec, è essenzialmente un plugin di decodifica progettato per Avisynt capace di decodificare ogni stream MPEG-1 e 2 leggibile da DGIndex. Utilità addizionali includono modifiche degli

spazi del colore, iDCT ottimizzate, processi di deblocking e deringing e filtri di luminosità.
AutoCrop: è un filtro di AVISynth che automaticamente croppa i bordi neri di un clip, opera sia in modo preview (dove sovrappone le informazioni per il cropping consigliato), che in modo cropping col quale taglia realmente il clip. Può inoltre garantire che larghezza ed altezza siano multipli di un valore specificato in modo tale che possa essere passato senza problemi al compressore video.
KernelDeint: Questo filtro deinterlaccia usando un approccio di tipo kernel. Nelle aree deinterlacciate, produce risoluzioni verticali di gran lunga migliori rispetto alla semplice eliminazione di un field.
RemoveGrain: è un filtro molto potente per la rimozione del rumore video, si presenta in 4 versioni differenti ottimizzate a seconda della tipologia del processore in uso. E' essenzialmente un filtro spaziale ma in combinazione con altri filtri appartenenti allo stesso package, Clense e Repair, è possibile definire delle funzioni Avisynth per operare in ambito spazio-temporale o su sorgenti interlaced. Ogni denoiser spaziale adotta più o meno la stessa tecnica, vale a dire che il pixel di riferimento viene stimato se essere rumore o meno, ed eventualmente rimpiazzato con uno dei pixel adiacenti, la stima viene fatta confrontando il luma, il chroma o entrambi.

3.4.3 Virtualdub

VirtualDub e, nel nostro caso, una versione modificata chiamata VirtualDubMod è un programma freeware ed opensource disponibile sui sistemi operativi Windows, per l'editing video lineare, e consente non solo di comprimere o ricomprimere un flusso video ed audio, ma anche di applicare un gran numero di filtri ed effetti nonché di catturare i vostri video dalle fonti più disparate. VirtualDubMod, grazie anche ad una perfetta compatibilità con AviSynth è in grado di lavorare su praticamente qualunque tipo di file video e di applicare un numero enorme di effetti (tanto che può essere utilizzato con successo anche per il "restauro" video); tramite AviSynth è possibile applicare filtri, importare formati, ridimensionare un file video prima ancora di aprirlo in VirtualDubMod, l'utilizzo di AviSynth, consente di anticipare le operazioni di cropping, ridimensionamento e filtraggio dell'eventuale video da processare, semplificando di molto la procedura da seguire in VirtualDubMod, nonché velocizzando le operazioni dato che è possibile lavorare direttamente nello spazio di colore YV12. VirtualDubMod comprende features molto avanzate, inclu-

de numerosi filtri e permette l'aggiunta di differenti tecniche di video-processing tramite l'aggiunta di plugin esterni. L'utilizzo principale del software consiste nella compressione audio e video, nella divisione o unione dei file video e nel multiplexing tra audio e video, ed include un gran numero di opzioni di cui indichiamo in seguito le principali:
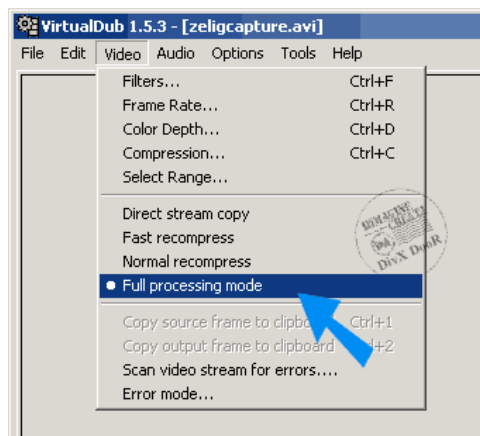
Immagine 3.7 - Menù video di Avisynt

- ***Video:*** questo menù include le varie opzioni selezionabili nella gestione dei flussi video, le parti più importanti includono la selezione del metodo di processing selezionato tra i seguenti:
  - *Direct Stream Copy:* Questa opzione consiste nella copia semplice dei flussi video, a cui non possono essere applicati filtri o algoritmi di compressione, utilizzato nelle operazione di divisione o unione dei files.
  - *Fast Recompress:* Questa opzione è utilizzata per le compressioni semplici e consiste nell'applicare le configurazioni più semplici ai vari algoritmi di compressione per rendere l'operazione più rapida possibile. Utilizzando questa modalità non è possibile accedere ai filtri del Software.
  - *Normal Recompress:* Questa opzione consiste nell'utilizzare le versioni standard degli algoritmi di compressione, permettendo una buona mediazione tra qualità e velocità; anche in questa modalità non è possibile utilizzare i filtri.
  - *Full Processing Mode:* Questa modalità permette di accedere a tutte le potenzialità di editing di VirtualDub, come l'applicazione di filtri, modifica degli spazi del colore e modifica del framerate. La seconda parte fondamentale nel processo di transcodifica include la selezione e la configurazione dell'algoritmo di codifica, nella finestra "Select video compression", il programma mostra sulla sinistra l'elenco di tutti i codec video installati nel sistema.
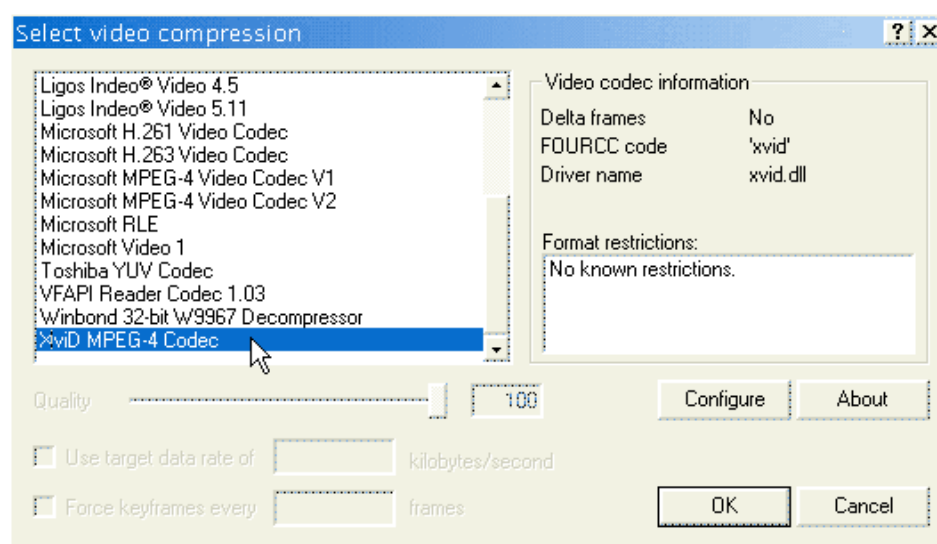
Immagine 3.8 - Elenco dei codec video di Avisynt

Selezioniamo in questo elenco il codec desiderato; una volta selezionato il codec desiderato premendo il tasto "Configure" che aprirà una nuova finestra, a seconda del Codec scelto, ci troveremo di fronte a diversi pannelli di configurazione, che presentano alcune impostazioni fondamentali comuni a tutti, e alcune regolazioni più complesse.

- ***Streams:*** La sezione audio in VirtualDubMod è ora compresa nella più generale sezione Streams ed include le opzioni di manipolazione e dei flussi esterni al video, in particolare l'audio; per una lista delle tracce disponibili, dalla finestra principale cliccate su "Streams-->Stream list":
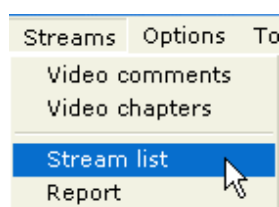
Immagine 3.9 - Menù stream di Avisynt

In questa sezione sono incluse sia le varie tracce audio i dati (generalmente sottotitoli) presenti nel file che si sta' processando o eventuali altri stream aggiunti in seguito. Cliccando sopra una delle tracce audio con il pulsante destro del mouse, compare un menù con diverse opzioni, per prima cosa si sceglie il metodo di processing dello stream (similmente nel caso del video); per mantenere inalterata la traccia nel suo formato originale si usa l'opzione "Direct stream copy" (ovvero copia diretta dello stream), selezionando invece "Full processing mode" si avranno a disposizione un gran numero di strumenti avanzati:

- *Interleaving:* opzione disponibile in entrambe le modalità può essere utilizzata per risolvere (ma anche prevenire) eventuali problemi di sincronizzazione tra audio e video, la parte più importante è la selezione del ritardo (positivo o negativo) da assegnare al flusso audio.
- *Conversion:* consente di modificare la frequenza di campionamento della traccia audio (Sampling Rate), effettuare una conversione stereo/mono e modificare la precisione con cui vengono memorizzati i dati (8 o 16 bit).
- *Volume:* permette di regolare l'amplificazione del segnale audio.
- *Compression:* consente di accedere alla lista dei filtri e codificatori ACM installati onde convertire l'audio in formati; tuttavia in genere, data la mancanza di codificatori di buon livello in formato ACM, non consiglio di effettuare la conversione tra formati con tale metodo, quanto piuttosto utilizzando software esterni creati appositamente ad hoc quali Besweet e/o HeadAC3he.
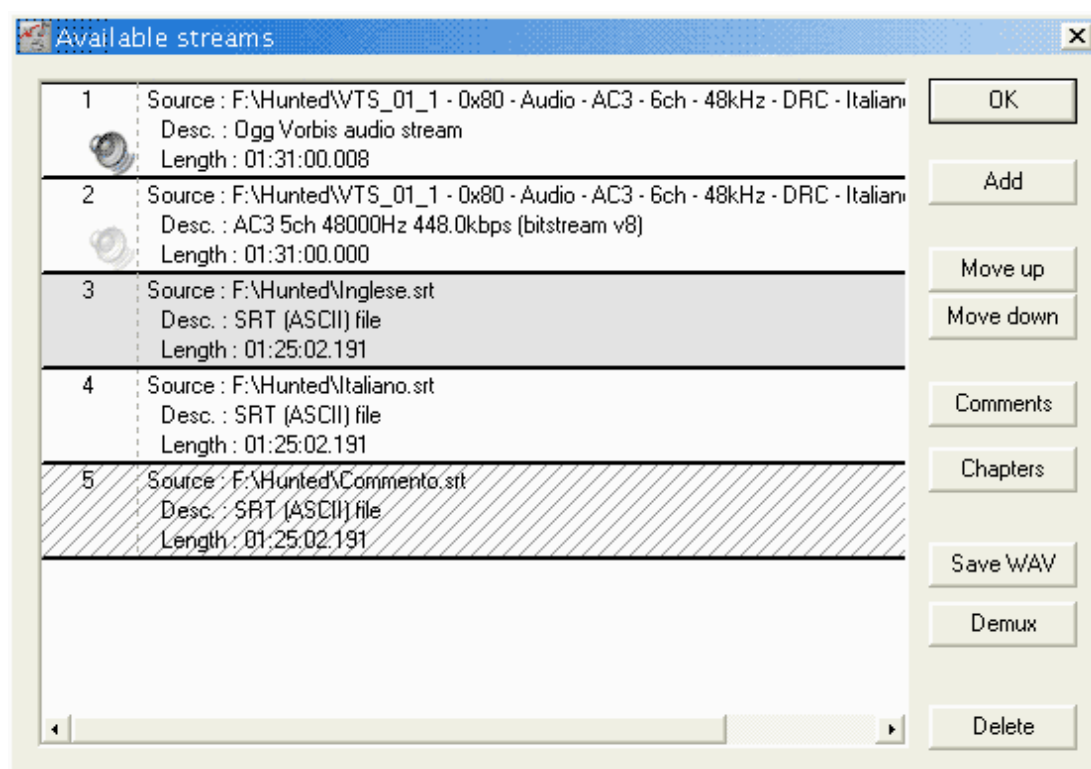
Immagine 3.10 - Elenco degli stream allegati ad un video in Avisynt

- *Use advanced filtering:* consente di abilitare l'accesso ad una opzione ulteriore "Filters", tramite la quale è possibile applicare un certo numero di effetti e filtri avanzati alla traccia audio (alla stregua della medesima opzione nel video)... ma ora siamo già al di fuori dello scopo di questa guida.

Anche per la gestione della parte audio disponiamo di due opzioni: possiamo infatti scegliere se lasciare l'audio del file avi invariato, e in questo caso selezioneremo

la voce "Direct stream copy" (vedi figura al lato), oppure possiamo comprimerlo, e in quest'altro caso selezioneremo la voce "Full Processing Mode". Un'altra importante opzione è l' ”Interleaving” che consiste nello strumento da utilizzare per la corretta sincronizzazione tra audio e video. Un altra importante caratteristica di VirtualDubMod è la presenza di un linguaggio di scripting interno che permette di configurare totalmente l'applicazione e di eseguire un qualsiasi tipo di processo in modo automatico.

3.5 Streaming Video

Il termine streaming identifica un flusso di dati audio/video trasmessi da una sorgente a una o più destinazioni tramite una rete telematica, sostanzialmente esistono due tipologie di streaming:

- ***Streaming on demand:*** i contenuti audio/video sono inizialmente compressi e memorizzati su un server come file, un utente può richiedere al server di inviargli i contenuti audio/video ma non è necessario scaricarli per intero sul PC per poterli riprodurre. I dati ricevuti vengono decompressi e riprodotti pochi secondi dopo l'inizio della ricezione, questo ritardo serve a creare un polmone per rimediare a ritardi o microinterruzioni della rete. Di questo tipo sono i flussi streaming di Real Video e Real Audio, Windows Media Player, QuickTime.
- ***Streaming live:*** simile alla tradizionale trasmissione radio o video in broadcast, anche in questo caso i dati sono trasmessi utilizzando opportune compressioni per alleggerire più possibile il carico sulla rete. La compressione dei contenuti introduce nel flusso un ritardo di circa dieci secondi.

3.5.1 Protocolli per lo streaming

Generalmente la trasmissione in streaming avviene utilizzando il protocollo RTP a livello di applicazione ma sono possibili anche altri protocolli, ad esempio il protocollo MMS () o il semplice HTTP. Protocolli a datagrammi come l'UDP inviano il flusso come una serie di pacchetti di piccole dimensioni, questo metodo è semplice ed efficiente tuttavia i pacchetti possono essere persi o corrotti lungo il tragitto, a seconda del protocollo e dell'estensione della perdita i pacchetti possono essere recuperati applicando tecniche di correzione degli errori o interpolazione dei dati. I protocolli *Real-time Streaming Protocol* (RTSP), *Real-time Transport Protocol* (RTP) e *Real-time Transport Control Protocol* (RTCP) sono disegnati specificatamente per lo streaming su reti, gli ultimi due sono costruiti sul protocollo di trasporto UDP. Protocolli affidabili, come il TCP, garantiscono la corretta consegna di cia-

scun bit nel flusso multimediale, tuttavia lo rendono possibile tramite sistemi di timeout e ritrasmissioni che li rendono più complessi da implementare; inoltre quando si verifica una perdita nei dati sulla rete lo streaming è interrotto per il tempo necessario al protocollo a localizzare la perdita e ritrasmettere i dati, sebbene tale effetto possa essere minimizzato tramite l'utilizzo di un buffer questo genere di comportamenti non è molto apprezzabile in un sistema id streaming, per questo motivo generalmente si preferisce l'utilizzo del protocollo UDP. La distribuzione dei dati di streaming può essere di tipo:

- ***Unicast:*** si utilizza una connessione client/server, viene inviata una copia separata dello streaming media dal server per ogni client, questo è il metodo più semplice ma porta ad una massiccia duplicazione dei dati sulla rete.
- ***Multicast:*** tale modello di servizio prevede che un calcolatore invii i pacchetti ad un indirizzo associato al gruppo multicast, molto efficient enel caso dello straming dato che il flusso viene inviato una sola volta ad un M-Router (Multicast Router) che provvederà alla duplicazione dei pacchetti solo per un tratto limitato; sfortunatamente questa modalità non è generalmente supportata dato che richiede una progettazione specifica.
- ***Peer to Peer:*** permette ai client che già hanno iniziato a ricevere iil flusso di ritrasmetterlo ad altri client; ciò previene un eccessiva duplicazione dei dati da parte del server ed impedisce alla sua sua connessione di rete di diventare un collo di bottiglia; questo permette una migliore distribuzione del traffico di rete ma richiede una progettazione più complessa.

Visto l'enorme successo avuto dallo streaming, ormai sono moltissimi i server costruiti appositamente per ospitare esclusivamente file audio o video.

3.5.2 VLC

VideoLAN è una completa soluzione software per lo streaming video, sviluppata dagli studenti dell' Ecole Centrale Paris, progettata per l'invio di video MPEG su reti a banda larga. La soluzione Video LAN include:

• VLS (VideoLAN Server): server di streaming per video MPEG, DVD, canali digitali satellitari o terrestri e video live sulla rete sia in unicast che in multicast.

• VLC (VideoLAN Client): molto conosciuto come video player è in grado di ricevere, decodificare e visualizzare molti tipi di contenuti multimediali sotto diversi sistemi operativi, nelle ultime implementazioni può anche sostituire VLS come server per l'invio di flussi video.

VLC media player (originariamente chiamato VideoLAN Client) è un media player che supporta molti codec audio e video, formati file come DVD, VCD e vari protocolli per lo streaming; può essere utilizzato anche come server per trasmettere stream in unicast o multicast su IPv4 o su IPv6 su un network a larga banda. Il media player impiega la libreria codec libavcodec del progetto FFmpeg per maneggiare molti dei formati supportati, ed utilizza la libreria di decriptazione DVD libdvdcss per gestire i playback dei DVD cifrati. VLC è uno dei media player più disponibile su varie piattaforme, infatti è disponibile per Linux, Microsoft Windows, Mac OS X, BeOS, BSD, Pocket PC, Solaris.
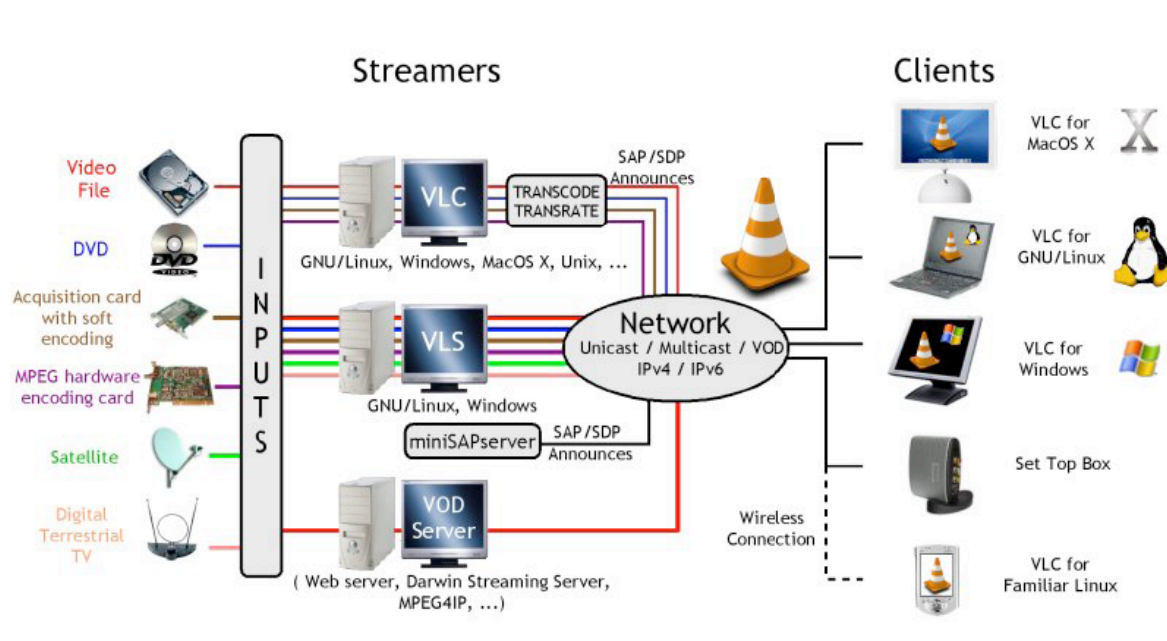
Immagine 3.11 - Overview delle soluzioni VideoLAN

VLC ha un design modulare che rende semplice l'inclusione di moduli per nuovi formati video, codecs o metodi di streaming; questi principi si estendono anche ad altre aree, ad esempio sono presenti un'ampia varietà di interfacce, output, controlli e filtri per audio e video. Ci sono più di 300 moduli in VLC, la GUI standard è basata su wxWidgets per Windows/Linux, Cocoa per MacOSX e Be API su BeOS; ed è simile per tutti i sistemi. VLC oltre che da media player può agire anche da streaming server, ed è dotato di interfacce di controllo remoto, la Remote Control Interface basata su testo che utilizza la sintassi dell'interfaccia a linea di comando, inoltre dispone di interfacce telnet ed HTTP. Le funzionalità di VLC includono:

- l'abilità di riprodurre contenuti video anche se incompleti, danneggiati o ancora in fase di download, questo perché è un player basato su pacchetti.

- la capacità di accedere a file iso così che l'utente possa riprodurre immagini disco, anche se il sistema operativo non è in grado di lavorare direttamente si tali immagini.
- supporta numerosi codec e formati inclusi MPEG-1, MPEG-2 e MPEG-4, file DivX e XviD, formati DVD e VCD, trasmissioni da satellite card (DVB-S) o acquisire filmeti da sorgenti esterne (Webcam o videocamere).
- permette la ricezione o l'invio di streaming di video in qualsiasi formato utilizzando diversi protocolli come RTP, HTTP, MMS o collegamento diretto con distrbuzione Unicast o Multicast in IPv4 o IPv6.
- include alcuni filtri che permettono la distorsione, rotazione, divisione, deinterlacciamento ed aggiunta di dati (ad esempio sottotitoli).

Capitolo 4

Architettura GRID

Nel campo delle architetture di calcolo parallelo, il Grid Computing rappresenta oggi, senza alcun dubbio, la frontiera delle attività di ricerca, tale fatto è testimoniato sia dal numero di progetti di ricerca dedicatigli, sia dal numero di grandi ditte di sistemi di calcolo, *software* e informatica in generale, che si sono dotate di una loro "soluzione Grid" (IBM, Sun, ecc). Il termine Grid computing (letteralmente, "calcolo a griglia") indica un'infrastruttura distribuita per consentire l'utilizzo di risorse di calcolo e di storage provenienti da un numero indistinto di calcolatori (anche e soprattutto di potenza non particolarmente elevata) interconnessi da una rete; il termine "griglia" deriva dalla similitudine fatta dai primi ideatori del Grid Computing secondo i quali in un prossimo futuro si sarebbe arrivati ad utilizzare le risorse di calcolo alla stessa stregua dell'energia elettrica, ovvero semplicemente attaccando una spina all'infrastruttura energetica, in inglese Power grid. L'idea del Grid computing, di cui recentemente si sente spesso parlare come la prossima rivoluzione dell'informatica, risale però a circa metà degli anni Novanta. Le 'griglie di calcolo' vengono prevalentemente utilizzate per risolvere problemi computazionali di larga scala in ambito scientifico e ingegneristico, sviluppatesi originariamente in seno alla fisica delle alte energie (in inglese HEP), il loro impiego è già da oggi esteso alla biologia, all'astronomia e in maniera minore anche ad altri settori. Una grid è in grado di fornire agli utenti di un gruppo scalabile senza una particolare caratterizzazione geografica (gergalmente detto VO ossia Virtual Organization) la potenzialità di accedere alla capacità di calcolo e di memoria di un sistema distribuito, garantendo un accesso coordinato e controllato alle risorse condivise e offrendo all'utente la visibilità di un unico sistema di calcolo logico cui sottomettere i propri job. L'idea del Grid computing è scaturita dalla constatazione che in media l'utilizzo delle risorse informatiche di una organizzazione è pari al 5% della sua reale potenzialità. Le risorse necessarie sarebbero messe a disposizione da varie entità in modo da creare un'organizzazione virtuale con a disposizione un'infra-

struttura migliore di quella che la singola entità potrebbe sostenere. In questo capitolo faremo una breve introduzione sulla XXX Grid per poi mostrare in linea generale l'architettura Avanade Grid presente nel Consorzio.

4.1 Grid Computing

Secondo la definizione del progetto Globus, uno dei primi, più importanti e, dal punto di vista dei finanziamenti ottenuti, più fortunati progetti relativi al Grid Computing: *"Le Grid sono ambienti persistenti che rendono possibile realizzare applicazioni software che integrino risorse di strumentazione, di visualizzazione, di calcolo e di informazione che provengono da domini amministrativi diversi e sono geograficamente distribuite"*. La definizione di Grid si basa, quindi, sul concetto di condivisione di risorse di diverse tipologie e si presta, quindi, a categorizzare diverse tipologie di Grid:

- ***Grid computazionali***: è l'aggregazione di risorse di calcolo provenienti da domini di sicurezza e gestione differenti, tale aggregazione è finalizzata a fornire a un insieme di utenti potenza di calcolo *on-demand*, in modo disaccoppiato dalla provenienza, cioè dai nodi che la stanno fisicamente fornendo. Si consideri, ad esempio, il caso di una società multinazionale con sedi sparse in tutto il mondo, ognuna di queste sedi possiede, oggi, un parco di architetture di calcolo dimensionato o secondo un *worst-case*, dimensionato cioè per soddisfare non le necessità medie ma i picchi di richiesta, oppure dimensionato per soddisfare le necessità medie e, quindi, non in grado di gestire le situazioni di picco. Una struttura Grid in grado di sopperire alle richieste di picco con potenza computazionale proveniente dalle altre sedi della società permetterebbe di dimensionare in modo più consono alle reali necessità quotidiane il parco macchine realizzando così un notevole risparmio, allo stesso tempo, lo stesso meccanismo di raccolta della potenza di calcolo sull'intero insieme delle risorse di calcolo della società permette di realizzare un supercalcolatore disponibile dinamicamente nel momento del bisogno.
- ***Grid di dati:*** possono essere considerate una delle forme evolutive del Web, infatti, come il Web, nascono per contenere grandi moli di dati distribuite in domini differenti per gestione e posizione geografica, a differenza del Web, dove, pur data la ricchezza delle informazioni presenti, mancano sia meccanismi espliciti di standardizzazione dei significati associati a queste informazioni, sia strumenti di elaborazione concorrente delle informazioni ottenute, le Grid di dati coniugano la

ricchezza delle sorgenti con gli strumenti adatti a far sì che le informazioni presenti possano essere facilmente correlate e vengano, quindi, a dotarsi di un alto valore aggiunto rispetto al mero contenuto. Si consideri, a titolo di esempio, il caso delle basi di dati contenenti i casi clinici dei singoli ospedali nel mondo, ognuno di queste basi di dati possiede, già presa singolarmente, un fortissimo valore in quanto permette di analizzare quali siano state nel passato le decisioni dei clinici. Tale valore però, aumenta in modo non lineare se, aggregando le basi di dati in una Grid di dati, si rende possibile analizzare, confrontare e correlare le decisioni prese da clinici di scuole diverse in presenza di patologie uguali o, quantomeno, comparabili.

- ***Grid di applicazioni e servizi:*** rappresentano una delle visioni più futuribili nel campo del Grid Computing, esse, infatti, non si limitano a essere una versione globalizzata all'intera Internet del concetto di *Application Service Provider* (ASP), cioè della possibilità di affittare un certo tempo di esecuzione di una specifica applicazione su di un *server* remoto, ma contemplano anche la aggregazione di componenti applicativi sviluppati indipendentemente al fine di realizzare nuove e originali applicazioni. Si consideri, a titolo di esempio, il seguente caso: la costruzione di una nave è un processo che ormai non coinvolge più il solo settore cantieristico in senso stretto, ma, al contrario, richiede l'intervento di un largo insieme di fornitori dedicati ai singoli sottosistemi, lo sviluppo di tali sottosistemi, come la parte motoristica, la parte di stabilizzazione o la parte dedicata alla sensoristica di navigazione, sono affidate esternamente all'ente cantieristico nominalmente responsabile della costruzione navale. In uno scenario tradizionale, ognuno di questi fornitori ha una visione limitata al sottosistema di cui è incaricato e si viene così a perdere la visione globale del prodotto finale sino al momento in cui, ottenuti tutti i singoli sottosistemi dai diversi fornitori, l'ente cantieristico sarà in grado di assemblare il prodotto finito. Questo non permette di valutare all'interno del *loop* di progettazione gli effetti dovuti alle interazioni tra i diversi sottosistemi, e rischia, quindi, di portare a scoprire indesiderati effetti collaterali solo in fase di collaudo dell'intero sistema. Una soluzione a questa *empasse* è ottenibile tramite l'integrazione in una Grid di applicazioni di elementi simulativi relativi ai diversi sottosistemi, tuttavia, elementi simulativi sviluppati secondo lo standard internazionale attuale, cioè (HLA) *High Level Architecture*, non supportano l'assemblaggio dinamico a *run-time* di un sistema completo, hanno un sup-

porto limitato per l'esecuzione distribuita e non permettono di schermare completamente agli altri utenti i dettagli interni al componente. Al contrario, una Grid di applicazioni deve permettere a ciascun fornitore:

- di mantenere il pieno controllo del componente della simulazione relativo al suo prodotto.
- di esporre solo un'interfaccia opaca.
- di assemblare l'intero sistema nave in una simulazione completa.

In una situazione di questo genere, ovviamente, una Grid di applicazioni contiene al suo interno sia una Grid computazionale che una Grid di dati, le risorse di calcolo utilizzate per eseguire i diversi componenti applicativi sono aggregate dinamicamente a partire da domini di gestione diversi; contemporaneamente, i dati su cui si trova a operare l'applicazione complessivamente costruita provengono da sorgenti differenti e sono correlate tramite l'esecuzione coordinata delle componenti applicative.

- ***Grid di strumentazione*** consiste nella generalizzazione del concetto di accesso remoto ad apparati di costo elevato o per rendere fruibile in forma remota un *setup* sperimentale di elevato valore didattico; in una Grid di strumentazione, apparati gestiti da enti diversi possono essere integrati per aggregare esperimenti complessi non possibili in nessuno dei singoli siti coinvolti nella Grid o per condividere strumentazione didattica in modo da rendere disponibile a corsi di studi afferenti ad atenei diversi una struttura condivisa per la realizzazione di esperimenti di comprovato valore didattico.

4.2 Avanade Grid

4.2.1 Architettura

La figura sottostante mostra la visione concettuale dell'architettura Avanade Grid e delle sue maggiori componenti, di seguito è fornita una descrizione dei ruoli e delle responsabilità dei componenti principali.

4.2.1.1 External Client

Dal punto di vista dell'Architettura Avanade Grid un *client esterno* è un sistema (indipendentemente dal genere) da cui è inviata una richiesta di esecuzione di task; la sottomissione della richiesta può avvenire direttamente, tramite un programma esterno o eseguendo un altro task per scenari che richiedono la creazione di subtask. La richiesta viene eseguita tramite un'istanza della classe *TaskExecutionRequest*, al cui interno sono include tutte le informazioni necessarie alla cor-

retta gestione del task, che viene sottomessa allo sheduler tramite la *scheduler client interface*; la richiesta del task è generalmente accompagnata da una serie di dati che forniscono gli input necessari alla corretta esecuzione del processo.
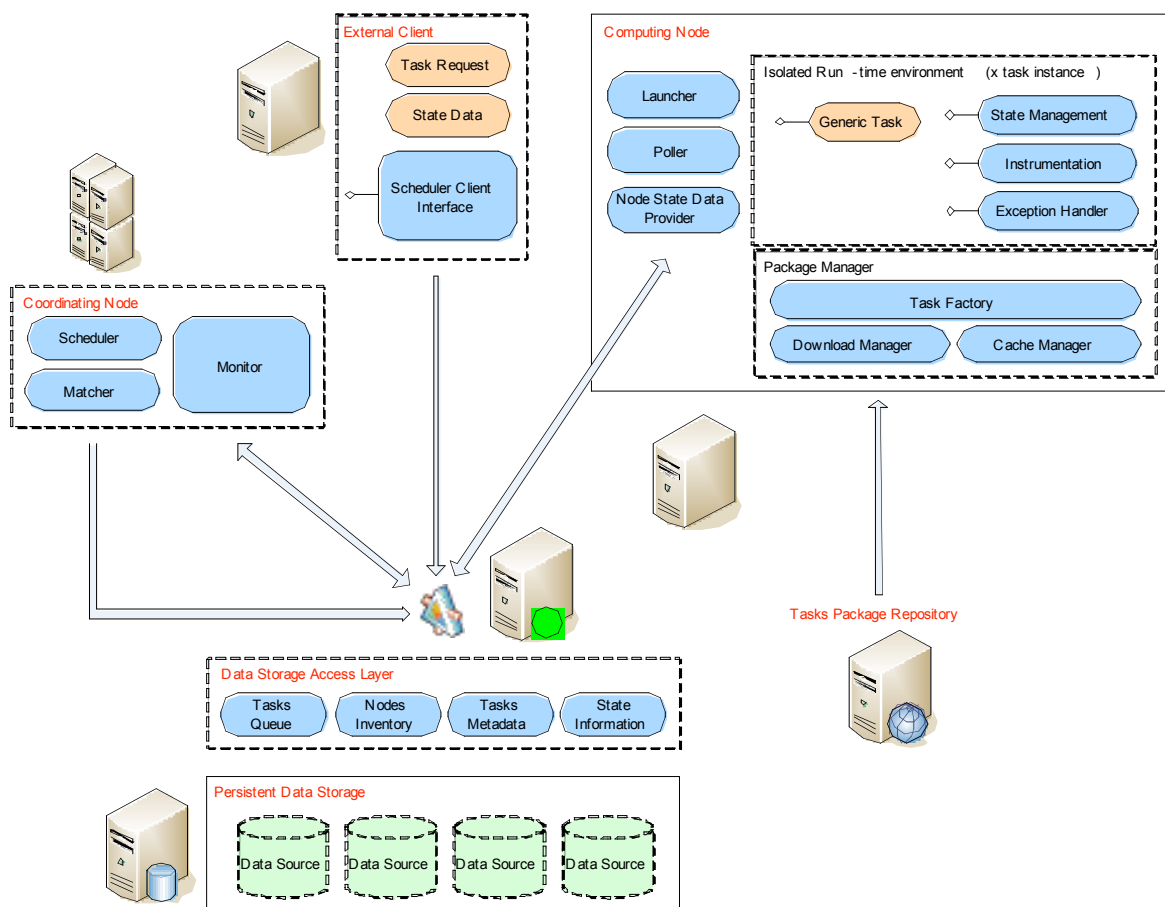
Immagine 4.1- Architettura Avanade GRID

Tali dati sono salvati in un contenitore general purpose chiamato *StateData* che è memorizzato nel *persistent data storage* (tipicamente un database) utilizzato per conservare e condividere le informazioni durante l'esecuzione, gli oggetti collezionati nello StateData devono essere serializzabili in modo che possano essere salvati nel database esterno. Lo fornisce il collegamento tra il client ed il resto dell'architettura, esso accetta le richieste di task in arrivo ed esegue le azioni:

- aggiunge un task alla coda in base alle informazioni contenute nella richiesta (ad esempio il tipo del task).
- estrae il contenuto dello StateData associato al task e li salva nel persistent data storage tramite il *data storage access layer* che rende l'interfaccia client trasparente rispetto alle specifiche tecnologie utilizzate.

4.2.1.2 Coordinating Node

Ogni implementazione dell'Architettura Avanade Grid richiede almeno un coordinating node per supportare i processi di scheduler e monitor, tipicamente collocati, i componenti del nodo sono i seguenti:

- ***Scheduler:*** esegue il polling della coda dei task ad una frequenza prefissata cercando i task in attesa di essere eseguiti da un nodo valido, quando viene trovato un tale processo lo scheduler fa partire l'esecuzione di un algoritmo di matching specificato per il tipo di task in attesa responsabile di trovare il migliore assegnamento possibile tra i task in attesa ed i nodi disponibili. L'assegnamento deve essere realizzato specificatamente per fit al meglio i bisogni dei task in base alle informazioni disponibili sullo stato dei processi e l'hardware dei sistemi e si basa su diversi fattori.
- ***Monitor:*** Il monitor è il processo responsabile per l'osservazione costante dell'insieme dei task, verificando periodicamente il loro status in modo da assicurarsi che le cose procedano secondo le aspettative; in tal modo ci si assicura che i task siano effettivamente eseguiti dai computing node a cui sono assegnati in un ragionevole periodo di tempo. Il monitor esegue numerose verifiche, di seguito specifichiamo le principali:
  - i task assegnati devono essere accettati dai nodi responsabili dell'esecuzione in un tempo specificato.
  - i task devono essere eseguiti entro un tempo massimo specificato dal tipi di task.
  - il tempo passato tra due checkpoint all'interno di un task (tipicamente un task lungo) non deve eccedere un tempo specificato.
  - ogni computing node deve inviare un segnale di polling (chiamato *heartbeat*) ad una frequenza specificata.
  - se le verifiche non hanno successo il monitor interviene per evitare problemi di overall computing system, ad esempio annullando l'assegnazione di un task e rimandandolo in coda se si verifica un timeout, o escludendo un nodo dall'insieme dei computing nodes se non invia il suo heartbeat.
- ***Matcher:*** è il componente che si occupa di calcolare effettivamente il migliore assegnamento per un particolare task tramite l'algoritmo di matching specificato per quel tipo di task.

- ***Node Information Storage:*** è una classe configurabile responsabile per la gestione delle informazioni riguardanti i nodi e i task che vi sono stati assegnati, l'accesso a queste informazione è previsto tramite il storage centralizzato; questa classe compie le seguenti operazioni:
  - controllo della sintassi delle richieste del task, fornisce un aiuto durante l'inserimento dei dati per il fatto che ciascun tipo di *Node Information* può richiedere una diversa sintassi.
  - creazione della lista dei nodi che soddisfano la particolare richiesta.
  - salvataggio del task con la proprie informazioni.
  - salvataggio di un particolare nodo con le proprie informazioni.
  - lettura dei dati di un particolare nodo.

4.2.1.3 Algoritmo di Matching

Il sistema di matching ha come obiettivo la selezione di un insieme di macchine in grado di eseguire i task, per ottenere questo risultato ogni tipo di task specifica un manifesto in cui sono specificate tutte le richieste per il completamento del processo, il manifesto consiste in un documento strutturato in sessioni che specifica gli attributi necessari che possono suddividersi in attributi statici o dinamici; i primi relativi alla configurazione hardware che i nodi devono soddisfare ed includono il clock della CPU, la memoria totale, la dimensione dei dischi ecc; i secondi descrivono lo stato del computing node necessario per l'accettazione di nuovi task ad esempio la percentuale di CPU disponibile, la memoria libera, ecc, parametri che cambiano a seconda del carico di lavoro presente sul nodo. Tutti questi attributi sono forniti dal sistema WMI presente nativamente in ogni nodo MS, il servizio grid che gira localmente esegue il polling verso il provider WMI locale per ottenere i vari attributi ed invia tutte queste informazioni al servizio di scheduler, ciò avviene tramite un meccanismo di sottoscrizione guidato da eventi che consistono nel raggiungimento di valori soglia in specifici valori; lo stesso metodo è utilizzato per la conservazione e l'aggiornamento delle informazioni raccolte dai nodi client e mantenute in un catalog. Il sistema di scheduling identifica le potenziali macchine target eseguendo delle query sul catalog con le specifiche richieste nel manifest del task in modo che l'insieme dei nodi così ottenuto possa essere ordinato in base ad un valore di ranking che permette di selezionare la macchina adatta dal punto di vista della piattaforma (per assicurarsi che una macchina possa effettivamente eseguire il task), ed il miglior nodo dal punto di vista delle performance, il rank di

un nodo si ottiene calcolando la *fitting measure* (misura dell'adeguatezza del nodo) definendo una metrica adeguata nello spazio degli attributi, un esempio è il seguente, dati i parametri:

Mr -> memoria richiesta
Mf -> memoria libera
Cr -> percentuale di CPU richiesta
Cf -> percentuale di CPU libera

da che si ottiene la seguente coppia di attributi:

{ (Mf – Mr) / Mf ; (Cf – Cr) / Cf } / Sqrt (2)

Il ranking è il valore normale della precedente coppia, più è vicino ad uno migliore sarà il nodo. Inoltre i nodi devono anche essere facilmente raggiungibili, per questo motivo nello scheduling si devono considerare anche le performance della rete. Se i task da schedulare sono più dei nodi selezionati il servizio può eseguire due azioni:

Mettere in coda i task rimanenti, per far ciò è necessario valutare i nuovi valori effettivi degli attributi utilizzati, rispetto all'esempio precedente si ottengono

Mef = Mf – Mr - > effective free memory
Cef = Cf – Cr - > effective free CPU percentage

La versione generale della formula è:

I valori effettivi devono essere calcolati ad ogni ciclo di scheduling e memorizzati nel catalog. Il secondo metodo consiste nel far eseguire più task su una stessa macchina, riassegnando i compiti in base al nuovo stato delle macchine. Oltre al sistema di ranking gli attributi possono anche essere utilizzati per specificare delle restrizioni sull'esecuzione di un task (ad esempio per farlo eseguire ad un gruppo di nodi piuttosto che un altro); tali restrizioni possono essere definite sia per i tipi di task che per i singoli task, e nel caso siano definite sullo stesso attributo le seconde sovrascrivono le prime.

4.2.1.4 Computing Node

Un computing node è una macchina generica su cui viene eseguito uno specifico servizio che compie un controllo periodico, a frequenza modificabile, della coda dei task centralizzata verificando la presenza o meno di task assegnati ad esso dallo scheduler; il controllo avviene invocando un'interfaccia che espone un insieme di API pubbliche. Se esiste un task del genere il nodo compie le seguenti azioni:

- verifica che i componenti binari (e tutti file generici) necessari all'esecuzione del tipo del task e del task specifico siano disponibili localmente, ed in caso contrario

ne effettua il download. La lista dei componenti necessari e della locazione relativa da cui è possibile effettuare il download (tramite HTTP o SMB) si ottiene eseguendo una query al catalogo centrale che contiene tutti gli attributi di configurazione per ogni tipo di task definito.

- una volta ottenuti i requisiti (in termini dei componenti necessari) il computing node istanzia le classi contenenti l'implementazione del task in base alla conoscenza del nome completo del task. Se alla sottomissione del task è compreso uno StateData esso viene recuperato dal depository centrale, deserialzzato ed iniettato nell'istanza appena creata. Per assicurarsi un ambiente di esecuzione isolato per ciascun task viene creato un nuovo AppDomain propriamente configurato per ospitare l'istanza.-
- infine il nodo avvia l'esecuzione del task invocando il metodo Run, ciò genera la creazione di un nuovo thread dedicato con una priorità che si accorda con la configurazione specificata per il tipo di task.

Il computing node è anche responsabile per l'esecuzione periodica dell'aggiornamento centrale dell'insieme dei parametri e degli attributi sia statici che dinamici relativi allo status del task durante l'esecuzione, ed al completamento, per assicurare che lo scheduler possa contare su informazioni adeguate su cui basare le decisioni riguardo la corretta assegnazione dei task e per riportare le eccezioni generate durante il processo al sistema centrale; anche queste operazioni si servono di API esposte da un servizio d'interfaccia piuttosto che tramite un accesso diretto al database. I componenti del nodo sono i seguenti:

- ***Poller:*** il poller è il componente che osserva la coda dei task ad una frequenza prefissata e configurabile alla ricerca dei task assegnati al nodo, quando tali task esistono il poller istruisce il launcher per l'attivazione dei task; è anche il componente responsabile della comunicazione con il monitor per la notifica del successo della procedura.
- ***Launcher:*** il launcher espone un interfaccia generica invocata dal poller per istanziare ed eseguire il task assegnato al nodo, quest'interfaccia è completamente astratta ed incapsula i dettagli implementativi dello start-up dei task e la strategia utilizzata per fornire ai task un ambiente di esecuzione isolato; le operazioni che il launcher esegue includono:
  - fornire al task factory di abbastanza dettagli per creare il corretto tipo e versione dell'istanza del task.

- forniscono all'istanza del task appena creata tutte le necessarie informazioni invocando l'apposito metodo di inizializzazione dell'interfaccia generica dei task.
- ***Node State Data Provider:*** è il componente responsabile per l'aggiornamento periodico dell'insieme dei parametri relativi allo stato del nodo permettendo al matcher di effettuare le decisioni ottimali riguardo l'assegnamento, questo è ottenuto utilizzando il supporto WMI fornito dal computing node.

Le seguenti componenti assieme compongono il package manager, lo strumento che si occupa della creazione e della gestione degli strumenti necessari all'esecuzione dei task

- ***Task Factory:*** il task factory è responsabile della creazione delle istanze del task basato sul tipo richiesto, il factory esegue una query al gestore della cache per verificare la disponibilità del package contenente l'implementazione del task; se la verifica ha successo viene utilizzata la versione presente nella cache, altrimenti viene invocato il download manager per recuperare il package dalla locazione specificata.
- ***Download Manager:*** per far eseguire un qualsiasi tipo di task su un computing node deve essere fornito un package contenente tutti i componenti necessari (inclusi gli assembly e i file di configurazione), il download manager compie lo scaricamento, la validazione e l'installazione di tali componenti sul nodo. La locazione da cui deve essere effettuato il download deve essere specificata per ogni tipo di task come parte dei suoi dati di configurazione ed il download può essere eseguito sia on-demand (ciò avviene quando il task è istanziato dal task factory ed il package relativo non è disponibile localmente) o tramite un servizio in background che verifica la presenza di package aggiornati da utilizzare.
- ***Cache Manager:*** per minimizzare il traffico di rete e migliorare l'efficienza dell'esecuzione sono fornite alcune capacità di chaching al computing node, due necessità principali:
  - evitare download non necessari dei package relativi ai task assegnati al nodo ogni volta che sono eseguite più istanze dello stesso tipo di task.
  - conservare i dati utilizzati più di frequente senza la necessità di accedere al database.

Il task è un pezzo di applicazione specifica che è eseguita all'interno di un ambiente di esecuzione isolato fornito dal computing node, l'implementazione della logica

deve aderire a delle specifiche di progettazione che assicurino la propria interoperabilità tra il task e l'ambiente di esecuzione; tale ambiente è dotato di alcuni strumenti utilizzati per facilitare l'esecuzione dei task.

- ***State Management:*** l'architettura Avanade Grid fornisce l'implementazione dei task con un'infrastruttura per la gestione dello stato per salvare e recuperare i dati necessari per l'esecuzione del task; l'infrastruttura è anche capace di rinfrescare gli StateData comunicando con il persistent data storage, questo comportamento è particolarmente utile nel caso di task a lunga esecuzione che richiedono il loro stato salvato periodicamente a dei ben definiti checkpoints che possono fornire un punto di accesso per riprendere l'esecuzione nel caso si verifichino condizioni anomale durante il processo.
- ***Instrumentation:*** l'utilizzo degli strumenti forniti dai componenti che prende parte in un sistema distribuito complesso è critica per raggiungere una buona comprensione di come le cose funzionino o, cosa anche più importante, del perché non funzionano. Tutti i componenti dell'architettura possiedono degli strumenti precostituiti con l'opportunità di aggiustare la granularità delle informazioni che il componente può esternare, inoltre l'architettura fornisce il supporto per aggiungere strumenti personalizzati ai task, principalmente ciò può essere raggiunto in due modi:
  - applicando attributi personalizzati specifici alle classi dei task
  - invocando un'interfaccia della strumentazione all'interno dell'implementazione del task.
- ***Exception Handler:*** l'architettura fornisce un framework per la gestione delle eccezioni che si verifichino durante l'esecuzione del task; è possibile fornire una gestione delle eccezioni specifiche per il tipo di task al livello della progettazione del task stesso altrimenti ci si serve del gestore ad alto livello che funziona con una prospettiva centralizzata; l'ambiente di run-time è il responsabile della cattura delle eccezioni e dell'invocazione del corretto gestore (il più specifico viene generalmente invocato per primo).

4.2.1.5 Data Storage Access Layer

Tutte le informazioni rilevanti per un sistema distribuito (come i task da eseguire e i parametri relativi ai computing node) secondo le specifiche dell'architettura Avanade GRID sono memorizzate nello storage persistente centralizzato; l'accesso alle informazioni contenute avviene tramite un layer che permette all'utente un ac-

cesso trasparente rispetto ai dettagli implementativi e le specifiche tecnologiche del database e i particolari schemi di dati utilizzati. Tale strato espone un interfaccia simile ad un web service, ed è costituito da diversi componenti ciascuno responsabile di fornire un tipo di dato differente, di seguito ne diamo una breve descrizione:

- ***Task Queue:*** questo componente è utilizzata da diversi tipi di consumatori (scheduler, monitor e computing node) e permette di accedere e manipolare la coda che contiene i task in esecuzione; a seconda dell'utilizzatore compie diversi tipi di operazione rispetto a questo componente, in particolare:
  - lo scheduler inserisce in coda le istanze dei task ed aggiorna le informazioni per notificare l'assegnamento ad uno specifico computing node e l'assegnamento ai task del loro Queue ID.
  - il monitor compie delle verifiche sullo status dei task per essere sicuri che sia ancora vivo e non abbia incontrato problemi di esecuzione.
  - il computing node controlla la coda per cercare dei task assegnati ad esso, inoltre compie l'update dello status dei task ad ogni cambiamento che si verifica durante l'esecuzione del processo (come l'accettazione del task, l'inizio o la terminazione del processo o il raggiungimento di un checkpoint)
  - L'implementazione della coda si serve di una tabella di un databaase SQL-Server come strumento di conservazione dei dati, ogni task viene mappato in una tupla contenente tutti gli attributi necessari; questa soluzione permette di raggiungere delle buone performance, e fornisce le necessaria flessibilità in termini di manipolazione della coda di economa rispetto all'infrastruttura.
- ***Nodes Inventory:*** questo componente è utilizzato per l'accesso e l'aggiornamento delle informazioni relative alla configurazione hardware e software dei nodi di computazione, rilevanti per eseguire al meglio il processo di matching tra i nodi e i task, i consumatori principali di tale componenti sono:
  - i computing node aggiornano periodicamente i dati loro relativi.
  - lo scheduler che accede alle informazioni dei nodi che serviranno per l'esecuzione dell'algoritmo di matching.
- ***Task Metadata:*** questo componente è utilizato per l'accesso e la modifica di informazioni relative ai differenti tipi di task ed alle loro configurazioni; a questo componente accedono prncipalmente:

- il configuration tool che gestisce le informazioni riguardo la configurazione e definizione dei task.
  - il computing node che prende i dettagli relativi ai package contenenti l'implementazione dei task.
  - lo scheduler che deve conoscere quale algoritmo di matching deve essere utilizzato per uno specifico tipo di task.
- ***State Information:*** questo componente è utilizzato per recuperare o aggiornare l'informazione dello stato dei task necessario per la corretta gestione delle esecuzioni; il componente che si occupa di aggiornare queste informazioni è soprattutto il computing node che esegue i task.

4.2.1.6 Persistent Data Storage

Dal punto di vista dell'architettura grid il persistent data storage è lo stato fisico che contiene i dati utilizzati da tutti gli altri componenti del sistema, l'approccio seguito consiste nell'utilizzare un insieme di database progettati specificatamente per conservare le informazioni necessarie; le ragioni per preferire la tecnologia dei database riguardano una maggiore flessibilità in termini di accesso e manipolazione dei dati e per maggiori performance e scalabilità.

4.2.2 Sviluppo su grid

Il task è visto come l'unità lavorativa di base che è chiaramente identificata all'interno di un programma complesso come un componente implementato in modo separato e auto contenuto; i task che compongono i programmi devono essere eseguibili su di un insieme di macchine. I tipi di task definiti per un programma sono qualificati da un insieme di attributi che li definiscono completamente, la lista degli attributi include:

- ***Nome:*** un nome unico e significativo che serve a riferirsi al task.
- ***Descrizione:*** una breve descrizione delle operazioni eseguite dal task.
- ***UniqueID:*** l'identificatore univoco del task.
- ***QueueID:*** un ID che identifica univocamente il task all'interno della coda, è creato assegnando un ID a ciascuna nuova coda ed incrementandolo ad ogni sottomissione di task.
- ***JobID:*** un ID che identifica ogni nuova istanza di un task.
- ***Priorità:*** la priorità assegnata al thread che esegue l'istanza del task.
- ***Livello di Priorità:*** la priorità dei task da eseguire assegnata dallo scheduler (Low, Medium, High)

- ***Package:*** l'insieme dei componenti necessari all'esecuzione dei task.
- ***Download Location:*** l'URL della locazione in cui si trovano i componenti da scaricare.
- ***Task Entry Point:*** il nome della classe che implementa l'interfaccia standard del task, i metodi di quest'interfaccia devono essere invocati per far partire l'istanza del task e controllare la sua esecuzione.
- ***Matching Algorithm:*** l'algoritmo utilizzato per il ranking dei nodi disponibili e la scelta del nodo migliore.

Un tool di configurazione specifico aiuta gli amministratori a definire il task e tutti gli attributi necessari; i metadati di configurazione relativi al task vengono conservati in una specifica tabella del database.

4.2.2.1 Creazione di un Task

Dal punto di vista architetturale i task sono classi sviluppate sfruttando il framework Miscrosoft .NET che ereditano da un base comune (*Avanade.AGA.Client.Task*) esposta dal framework AGA (*Avanade Grid Architecture*) che contiene l'implementazione della logica necessaria all'esecuzione nella struttura distribuita, tali classi sono raggruppate assieme ed inviate ad un repository centrale, che sarà recuperato dal nodo computazionale al momento dell'esecuzione. Di seguito esponiamo un esempio di un semplice task utilizzato per il testing:

```
using System;
using Avanade.AGA.BaseClasses;
using Avanade.AGA.Client;
namespace TaskSample
{
	public class SimpleTask : Task
	{
		public SimpleTask(){}
		public override TaskExecutionStatus ExecuteImpl()
		{
			// Reads parameters provided when the task was scheduled
			Int32 iNum1 = (Int32) Context.InputParameters["Num1"];
			Int32 iNum2 = (Int32) Context.InputParameters["Num2"];
			Int32 iTot = iNum1 + iNum2;
			// Saves the result of the processing to the task's state
			Context.State["Tot"] = iTot;
			// Commits the context
			Context.CommitState();
			// Traces a message
			LogTaskMessage(MessageType.TRACE, "Done!");
			// Signals task's completion
			return TaskExecutionStatus.Completed;
		}
	}
}
```

Come già detto si nota il fatto che la classe eredita dall' *Avanade.AGA.Client.Task* che assicura la corretta gestione del task e permette agli sviluppatori di utilizzare un insieme di strumenti disponibili e necessari per il compimento delle tipiche attività. Il nucleo dell'implementazione consiste nell'overriding del metodo ExecuteImpl nella classe base, tale metodo restituisce un *Avanade.BaseClasses. TaskExecutionStatus* un'enumerazione contenente i valori che rappresentano tutti i possibili valori di uscita di un task (tipicamente *completed* quando l'esecuzione termina con successo o *error* quando si è verificata una terminazione anonima); un altro componente importante è il *TaskExecutionContext* che permette l'accesso alle informazioni inerenti il contesto del task correntemente in esecuzione tramite la proprietà *Context* della classe *Task*, tale strumento fornisce numerose feature:

- ***Leggere i parametri in Input:*** la proprietà fornisce l'accesso allo *StateData* che contiene la collezione di sola lettura *InputParameter* contenete i dati in ingresso che possono essere forniti al momento della sottomissione del task o successivamente tramite il monitoring tool del framework.
- ***Leggere e scrivere informazioni di stato:*** il context espone una collezione che permette il recupero e la modifica delle informazioni sullo stato dallo storage.
- ***Sottomettere task annidati alla coda:*** in maniera simile alla creazione di thread annidati durante un processo può essere necessaria la creazione di sotto-task, il context fornisce gli strumenti per la gestione di tali strumenti; di seguito un esempio di comandi:

```
// SubTaks collection need to be initialized first for clean-up
Context.SubTasks.Initialize();
// Creates a new task's instance
TaskExecutionRequest tRequest = TaskRequestFactory.CreateTask("SimpleTask");
// Appends the sub task to the collection
Context.AppendSubTask(tRequest);
```

- ***Riportare le Eccezioni:*** la cattura delle eccezioni durante l'esecuzione avviene tramite uno stack che verrà poi salvato al momento del commit.

```
Context.AddException(new Exception("Unexpected error."));
Context.CommitExceptions();
```

- ***Specificare un Checkpoint:*** i checkpoint sono dei punti nell'esecuzione del processo da cui la stessa può essere ripresa in caso di interruzione senza bisogno di ripartire dall'inizio del task; tale obiettivo si ottiene salvando le informazioni rilevanti per il mantenimento dello stato del task e l'assegnazione della proprietà CheckPointID che permette l'identificazione del punto da cui ripartire.

```
// Save state information
Context.State["Data"] = objData;
```

```
// Setting checkpoint id
Context.CheckPointID = "CheckPointLabel";
// Commit the changes
Context.Commit();
```

- ***Modificare le Priorità:*** il context permette di modificare i parametri del task relativi sia alla priorità dell'esecuzione del task nel nodo (*Priority*) sia di quella relativa allo scheduling della coda (*PriorityLevel*).

```
tRequest.Priority = PriorityClass.Normal;
tRequest.PriorityLevel = PriorityLevelClass.High;
```

- ***Eseguire il Commit:*** il cambiamento delle proprietà del context al momento del commit è salvato nel persistent storage, può essere invocato il metodo globale ma sono fornite anche metodi specifici per il commit parziale in particolare *CommitState*, *CommitExceptions* o *CommitStatus*.

4.2.2.2 Deploying a Task

Una volta che l'implementazione del task è stata compilata in un assembly è necessario che esso sia reso disponibile ai nodi della griglia; il framework AGA .NET deliver un deployement automatico di tali assemblyes e di eventuali files addizionali raggruppati assieme in package fornendo ai nodi la possibilità di scaricarli da un repository web centrale al momento dell'esecuzione di un nuovo task. Le operazioni di gestione e configurazione dei package sono eseguite tramite la sezione *Package List* della console di monitoraggio disponibile nell'architettura, qui sono definite le proprietà del package:

- ***PackageID:*** l'identificatore univoco del package.
- ***Description:*** descrizione del package
- ***RootLocation:*** URL da cui il package è disponibile.
- ***Entry Point:*** nome completo delle classi e degli assembly che contengono l'implementazione del task.
- ***Priority Class:*** la priorità di base dei processi che ospitano il task.

Una volta che il package è stato definito è necessario configurare la versione del package, ciò è necessario perché l'invocazione del package al momento dell'esecuzione deve specificare la versione utilizzata che rappresenta una specifica sotto-directory della root location, ciascuna di queste directory contiene i file necessari all'esecuzione ed un file *manifest* da scaricare per primo che contiene la lista dei file da scaricare ed eventualmente le operazioni da compiere per completare l'installazione del package.
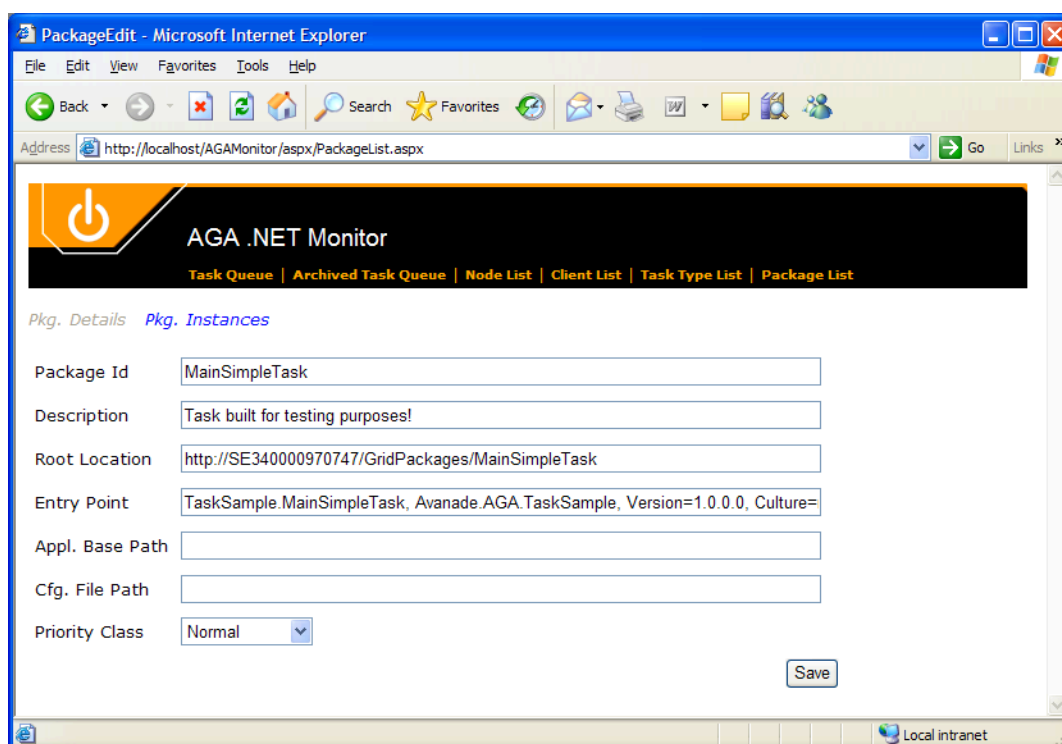
Immagine 4.2- Schermata Package List

4.2.2.3 Configurazione di un Task

Similmente alla configurazione di un package il monitor permette la configurazione dei task tramite il pannello *Task Type List* delle varie proprietà di un task

- ***TaskID:*** identificatore univoco del tipo tipo del task.
- ***Description:*** descrizione del tipo del task e delle operazione che compie.
- ***Startable:*** flag indicante se il task può essere avviato dalla console di monitoraggio.
- ***Package:*** versione del package associato al task.
- ***Running time:*** massimo tempo concesso all'esecuzione di un istanza di questo task, espresso in minuti.
- ***CheckPoint frequency:*** massimo periodo di tempo che può passare tra due checkpoint, espresso in minuti.
- ***Package transfer time:*** massimo intervallo di tempo concesso al download del package associato a questo tipo di task, espresso in minuti.
- ***WMI:*** espressione condizionale utilizzata per selezionare solo nodi che soddisfano specifiche condizioni sui parametri della macchina ottenuti tramite WMI.
- ***GROUPS:*** espressione condizionale utilizzata per selezionare per l'esecuzione solo nodi appartenenti ad uno specifico gruppo.

- ***Category restriction:*** espressione condizionale utilizzata per restringere ulteriormente la lista dei nodi in grado di eseguire il task in base a classificazioni personalizzate.

Immagine 4.3- Schermata Task Type List

- ***Rating formula:*** formula utilizzata per dall'algoritmo di matching per ordinare i nodo a seconda dei loro parametri.
- ***Task complexity:*** stima della complessità computazionale del task, utilizzata per verificare che ad un nodo non vengano assegnati dei task la cui complessità totale eccede le capacità del nodo.
- ***Maximum WaitingToBeAssignedTime:*** il massimo tempo in cui un task può rimanere nello status WaitingToBeAssigned prima di essere rimosso dalla coda, espresso in minuti.
- ***Scheduling Priority:*** definisce la priorità del task nella schedulazione.
- Le ultime due proprietà sono settate automaticamente al momento dell'inserimento dl task in coda.
- ***QueueID:*** identificatore del task all'interno della coda.
- ***JobID :*** identificatore univoco dell'istanza del task incrementato ad ogni creazione di un task dello stesso tipo.

4.2.2.4 Configurazione dei Computing Nodes

L'ultima fase consiste nella configurazione dei nodi che può essere compiuta tramite il pannello Node List, le informazioni necessarie sono le seguenti.

NodeId: nome DNS del nodo.

Immagine 4.4- Schermata Node List

Activity threshold: massima quantità di tempo concessa tra due heartbeat successivi utilizzati per identificare i nodi che non sono più attivi.

Max Task Complexity: limite superiore per la complessità totale dei task che possono enire assegnati al nodo.

Enabled: flag indicante se il nodo è disponibile o meno per la griglia.

Penalties: penalità assegnate al nodo come risultato di un fallimento nel portare a termine un task assegnatogli, i nodi che eccedono una certa soglia di penalità non sono più considerati disponibili per il matching.

Machine class: lista della classificazione personalizzata a cui appartiene il nodo.

Administrators e-mails: *indirizzi degli amministratori responsabili per il nodo.*

Oltre le informazioni descritte sono presenti una serie di informazioni aggiuntive che esplicitano le varie caratteristiche hardware, software e di configurazione del nodo.

Capitolo 5

Progetti Sviluppati

In questo capitolo parleremo dei due progetti sviluppati presso il laboratorio del Consorzio Operativo del Monte dei Paschi di Siena che trattano lo studio e la progettazione di dei semplici sistemi software che permettono la risoluzione di due problemi distinti; la semplificazione del procedimento di conversione video sfruttando un architettura distribuita ed il miglioramento della gestione del traffico di rete nell'utilizzo dello streaming video. Per ciascuno dei due sistemi forniremo una breve introduzione al problema ed al metodo utilizzato per risolverlo, mostreremo in dettaglio il funzionamento dei programmi, ed infine discuteremo dei risultati ottenuti rispetto ai modelli precedenti.

5.1 Architettura di Conversione Video

5.1.1 Approccio al problema

Il primo problema affrontato consiste nel trovare una soluzione al problema della creazione di una copia dei DVD-video, con il quale vengono solitamente effettuate le registrazioni dei video del Consorzio, in un file .AVI con protocollo di compressione MPEG-4 utilizzando un opportuno codec video, in particolare Xvid, per la memorizzazione su personal computer. Tale procedimento come già accennato è estremamente complesso e costoso in termini di complessità computazionale, la durata tipico di un'esecuzione del genere per un video di buona qualità in un formato a compressione maggiore può richiedere da 3 a 8 ore di intensivo uso del processore che diminuisce sensibilmente le prestazioni della macchina sui cui il processo è in esecuzione. L'idea alla base dell'applicazione sviluppata consiste nel parallelizzare il lavoro sfruttando l'architettura grid presente nel laboratorio per effettuare la distribuzione dei processi in modo semplice, automatico, ed altamente personalizzabile; tale metodo oltre a ridurre il tempo necessario al completamento del processo ha anche il beneficio di migliorare la gestione della sua complessità computazionale, dato che è possibile assegnare dinamicamente il lavoro a quelle macchine che al momento hanno un minore carico. In realtà il formato

standard per i sopraccitati processi di riproduzione video nel Consorzio è il WMV (Windows Media Video) ed il fine del processo consiste nel convertire il DVD in tale formato, purtroppo un metodo diretto di conversione non è disponibile al momento, perciò è necessario l'utilizzo di un formato intermedio, il già citato file avi, da cui eseguire una seconda transcodifica; il sistema sviluppato in questa tesi si occupa della prima parte del processo, la più facile da gestire in parallelo e personalizzare, dato l'elevato numero di programmi di editing esistenti, mentre la creazione del file wmv a partire dal file avi finale viene eseguita semplicemente sfruttando il programma Windows media Encoder della Microsoft. In termini pratici il lavoro si è svolto su un DVD-video del consorzio su cui era presente il filmato di una conferenza, tale filmato è memorizzato in tre file vob, i primi due della dimensione di 1 GB e l'ultimo della dimensione di 18,7 MB per una durata complessiva di 1,08 ore, e sono state effettuate diverse prove comparando il sistema distribuito con il vecchio metodo utilizzato nel consorzio, che utilizza il software AutoGK (Auto Gordian Knot), una utility che semplifica in una sola interfaccia ed in semplici passaggi la creazione di una copia di DVD convertita, o la ricodifica di filmati AVI o MPEG4, il programma supporta come sorgente direttamente il DVD oppure i file in formato MPG2, oltre ai filmati AVI-MPEG4; per le varie prove sono è stato utilizzato il terzo files, il più adatto per via delle sue dimensioni ridotte; mentre per i test finali sono stati processati tutti e tre i files che compongono il filmato. Bisogna sottolineare come il programma creato, così come il programma utilizzato per le comparazioni, sono in grado di processare solo file vob non criptati, quindi nel caso si debbano copiare dei DVD protetti è necessario una fase di pre-processing in cui si effettua il ripping dell'informazione, cioè l'estrazione dei contenuti audio e video dal dvd sull'hard disc in chiaro.

5.1.2 Sistema Sviluppato

Il progetto consiste in due parti distinte, la prima è il programma centrale che ha il compito di gestire la suddivisione dei file vob nelle varie parti, la creazione dei thread che invocano il task grid ed infine l'unione dei file avi generati da ciascuno dei processi separati in un filmato unico; la seconda parte consiste nel processo di codifica vero e proprio, esso è fatto partire dai task grid sulle macchine scelte e lavora invocando una serie di software esterni che, opportunamente configurati, prendono in ingresso una porzione del video DVD e restituiscono in uscita un file avi contenente il video in formato XviD.

5.1.2.1 Processo Centrale

Per prima cosa descriviamo l'interfaccia dell'applicazione mostrata in figura e le sue componenti:
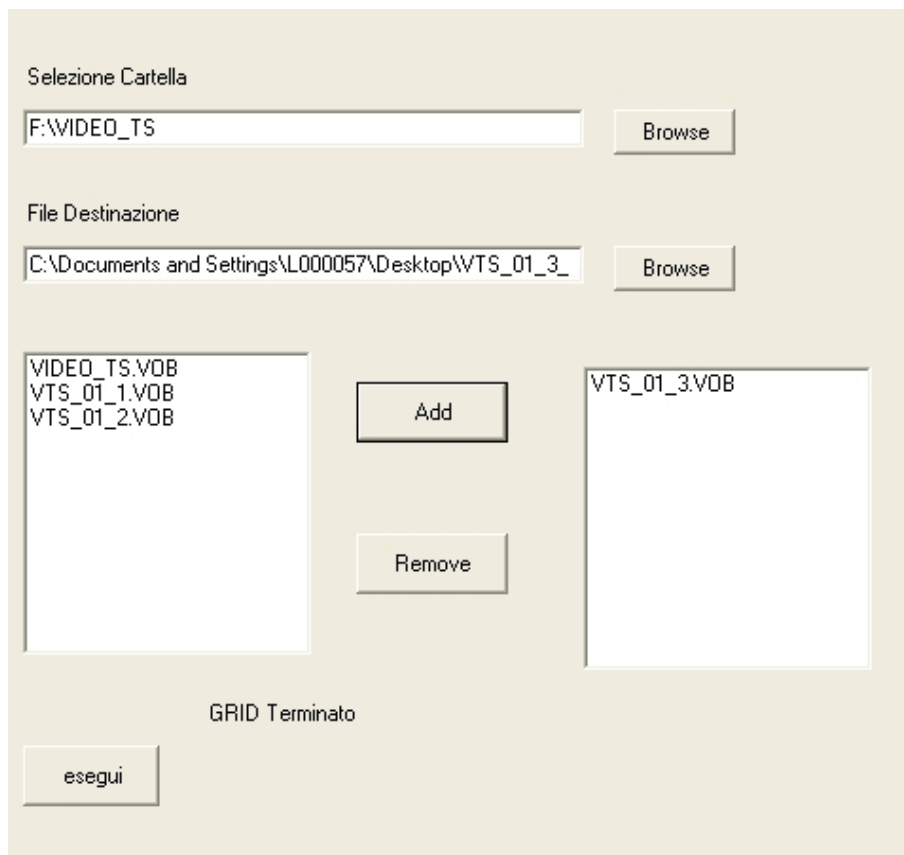
Immagine 5.1- Interfaccia dell'Applicazione

- ***Selezione Cartella:*** TextBox che contiene il nome della directory dome sono i file vob da processare.
- ***File Destinazione:*** nome e percorso del file di output in cui è salvato il filmato, se non specificato viene utilizzato il valore di default "C:\GRID\Test.avi"
- ***File Presenti:*** list box sulla sinistra, in questa sezione vengono mostrati tutti i file vob presenti nella directory selezionata, ciascuno di essi può a sua volta venire selezionato e mandato nella sezione accanto tramite il pulsante Add.
- ***File da Processare:*** listbox sulla destra, in questa sezione sono elencati i file vob selezionati per essere convertiti, l'ordine in cui essi sono mostrati rappresenta l'ordine in cui saranno memorizzati nel filmato finale, un file può essere tolto da questa lista tramite il pulsante Remove.
- ***Esegui:*** il pulsante che fa partire l'esecuzione del metodo principale di cui mostriamo una porzione di codice contenente le principali operazioni.

```
public void Esecuzione(){
[...]
foreach (string strtemp in listBox2.Items)
        {this.CutVob(folderBrowserDialog1.SelectedPath + "\\" + strtemp);}
[...]
DirectoryInfo dir = new DirectoryInfo(WORK_DIRECTORY);
FileInfo[] listaFiles = dir.GetFiles();
foreach (FileInfo fi in listaFiles)
        {[...]
        TextWriter command = File.CreateText(WORK_DIRECTORY +
                "\\lancioGRID_"+numFiles+".cmd");
  command.WriteLine("java   -cp   \".;Launcher.jar\"   com.avanade.aga.launcher.AGALauncher
        /Property:GridUnisi.properties /Task:TaskInteropNormal /Exec:Ripping.exe
                /Param:"+ fi.Name +" /Product:hpsipbase /User:root /TaskName:Prova
                /TaskPlatform:Windows /ExecDir:c:\\GRID");
        numFiles++;}
[...]
Thread[] chiamateGRID=new Thread[numFiles];
for(int i=0; i<numFiles; i++)
        {chiamateGRID[i] = new Thread(new ThreadStart(this.ChiamataGRID));
        chiamateGRID[i].Start();}
for(int i=0; i<numFiles; i++)
        {chiamateGRID[i].Join();}
this.UniteAVI();
[...]
```

La prima fase dell'esecuzione consiste nel parsing della listBox2, cioè la lista in cui sono indicati i file da processare, e per ciascuno di essi viene invocato il metodo CutVob(string filepath) il cui compito, come dice il nome stesso, consiste nella suddivisione del file vob che viene passato come argomento d'ingresso in una serie di files di dimensioni minori che saranno salvati nella directory di lavoro (definita tra le costanti dell'applicazione private const string WORK_DIRECTORY = "C:\\GRID\\Test";) trattasi di una cartella condivisa nella quale sono presenti tutti i componenti necessari al funzionamento dell'applicazione e dell'architettura grid invocata successivamente. La funzione di divisione dei files consiste in una semplice lettura del file in ingresso tramite la classe BinaryReader, e la successiva riscrittura di tali informazioni su due o più files in uscita utilizzando BinaryWriter; tale metodo semplifica l'operazione e risulta abbastanza efficace dato che i files vob consistono in stream MPEG-2 composti da pacchetti di 2048 bytes. A seconda delle dimensioni del file in ingresso sono state previste due modalità di divisione, se il file è minore di 500MB viene diviso in due parti, se la dimensione è superiore sarà diviso in quattro, in questo modo si ottengono file parziali la cui dimensione massima ha come limite superiore 250MB ciò permette un buon compromesso tra l'overhead generato dall'accodare e processare separatamente i file parziali e la complessità e durata del singolo processo. Nella seconda parte del metodo centrale vengono creati una serie di script contenenti i comandi grid necessari all'attivazione ed alla sottomissione dei task, esso consiste nell'esecuzione di un file jar (Launcher.jar), pre-

sente nella directory di lavoro che contiene tutti i file necessari al lancio del task grid, la sintassi del comando consiste nelle seguenti proprietà:

- ***com.avanade.aga.launcher.AGALauncher:*** classe invocata per far partire l'esecuzione dei task grid.
- ***/Property:*** il nome del file in cui sono specificate le proprietà del esecuzione del comando grid tra le quali sono specificati gli indirizzi dei vari server su cui sono attivi i servizi necessari (GridUnisi.properties), deve trovarsi nella stessa directory del file jar.
- ***/Task:*** il tipo di task utilizzato, nel nostro caso si usa il taskInterp che permette di lanciare tramite grid un eseguibile esterno.
- ***/Exec:*** il nome dell'eseguibile da lanciare sul nodo in cui è eseguito il task.
- ***/Param:*** i parametri da passare al programma esterno, per ogni parametro si usa una proprietà /Param distinta. Nel nostro caso il parametro necessario è il nome del file da convertire.
- ***/TaskName:*** il nome del task visualizzato nel monitor.
- ***/ExecDir:*** la directory in cui si trova l'eseguibile.

in seguito alla creazione di tali comandi vengono generati una serie di thread, uno per ciascuno script che invocano il metodo chiamateGRID, che si occupano di far partire l'esecuzione dei comandi lancioGRID; l'esecuzione di questi thread è bloccante per l'esecuzione del processo centrale che deve attendere la terminazione di ciascuno dei thread prima di poter proseguire, ciò per essere sicuri che ciascuna delle esecuzioni dei processi sui nodi esterni sia terminata e tutti i file da riunire siano disponibili. Dopo che i comandi sono partiti i task generati sono inseriti in coda, ed attendono di essere eseguiti da un nodo esterno; perché l'esecuzione vada a buon fine è necessario che in ogni nodo in cui il task può potenzialmente essere eseguito siano presenti il file eseguibile invocato dal task grid, così come tutte le varie applicazioni utilizzate da tale programma per eseguire le sue funzionalità, inoltre in ogni nodo deve essere presente la struttura della directory di lavoro in cui vengono salvate tutti i file di informazioni necessarie durante il procedimento, di seguito mostriamo un'immagine della coda dei task durante tale esecuzione.

Immagine 5.2- Task Queue

Una volta che tali task hanno terminato il loro lavoro nella directory di lavoro, o meglio nella sottodirectory /Test in cui sono memorizzati tutti i file temporanei generati dall'applicazione, sono salvati una serie di file .avi che contengono le porzioni del filmato convertite in XviD, da qui viene eseguita la parte finale del processo tramite l'invocazione dell'ultimo metodo, UniteAVI(), nel quale si esegue l'unione delle varie parti del filmato. Tale operazione viene effettuata tramite il software Virtualdubmod, creando uno script che contiene le operazioni e la configurazione del file necessarie; lo script è molto semplice consiste nell'aprire il primo file con il comando "VirtualDub.Open ("percorso file",0,0) ed in seguito ciascuno degli altri con il comando "VirtualDub.Append ("percorso file") essenziale è inserire tali comandi nello script nel giusto ordine in modo da rispettare l'ordinamento dei file e ricreare il filmato in modo corretto, per assicurarsi ciò nel momento in cui vengono suddivisi i file e creati i vari file .cmd viene creato un file listafiles.txt in cui sono scritti i vari spezzoni del video nell'ordine corretto. Le altre opzioni realmente importanti dello script consistono nello specificare il metodo di processing per il flusso video VirtualDub.video.SetMode(0), e per quello audio VirtualDub.stream [i].SetMode(0) ed eventuali stream dati come ad esempio una tracci aper i sottotitoli (vedi il 3° capitolo), ed infine la specifica dell'indirizzo del file in cui salvare tutto il lavoro svolto VirtualDub.SaveAVI("percorso file").

5.1.2.2 Applicazione su Nodi

Questa parte dell'architettura sviluppata esegue effettivamente il processo di conversione tramite l'invocazione di opportuni software esterni, e viene invocata dal task grid assegnato allo specifico nodo, di seguito mostriamo il comportamento di un task, inizialmente il task viene inserito in coda e attende l'assegnamento ad un

nodo, una volta eseguito il matching il task viene assegnato ad un nodo (stato *assigned*) una volta che il controller del computing node si accorge della presenza di un task assegnato esso è tolto dalla coda (*dequeued*) ed inviato in esecuzione; come si nota nel log del task è presente l'stdout del programma invocato dal task.

Task History Task State Task Parameters Task Constraints

Message Type	Message	Log Time	TaskRequestKey
Trace	Status changed to: 'WaitingToBeAssigned'	25/07/2007 11.53.25	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Trace	Status changed to: 'Assigned' on cl001000090003	25/07/2007 11.53.42	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Trace	Status changed to: 'Dequeued' on cl001000090003	25/07/2007 11.53.48	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Trace	Status changed to: 'Executing' on cl001000090003	25/07/2007 11.55.39	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Log	STD OUTPUT: C:\GRID\Test\VTS_01_1_4.VOB C:\GRID\Test\VTS_01_1_4.avi C:\\GRID\\Test\\VTS_01_1_4.avi 0	25/07/2007 12.21.14	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Trace	Status changed to: 'Completed' on cl001000090003	25/07/2007 12.21.14	3fb1d0a4-1873-4981-b557-1b6ab393aa77
Log	Task statistics: - User time: 139 seconds - Memory usage: 36236 KB	25/07/2007 12.21.20	3fb1d0a4-1873-4981-b557-1b6ab393aa77

Immagine 5.3- Storia dell'esecuzione di un task

In realtà tale parte consiste in due parti distinte, la prima consiste in un applicazione console che compie le prime fasi del procedimento, di cui di seguito mostriamo parte del codice del metodo main().

```
static void Main(string[] args){
[...]
        String file = "C:\\GRID\\Test\\"+args[0];
        String file_remote = "\\\\cl001000090003\\GRID\\Test\\"+args[0];
        File.Copy(file_remote,file,true);
        rip.Demux(file_remote);
        rip.ConversioneAudio1();
        String cammino = rip.ConversioneAudio2();
        int result=rip.Esecuzione(cammino,file_remote.Substring(0,file_remote.Length - 3)+"avi");
[...]}
```

La prima operazione compiuta dal processo consiste nell'effettuare la copia del file da processare dal nodo in cui si trova il processo centrale al nodo corrente, ciò è necessario perché l'accesso al file avviene tramite il file di indicizzazione che non è in grado di supportare un file remoto; in seguito vengono chiamate le funzioni che effettuano la conversione nel giusto ordine. La prima funzione invocata è De-

mux(“percorso del file video”), che si occupa di creare l’indice del file vob che gli viene passato in ingresso e di effettuare il demultiplexing tra il flusso audio e quello video; tale lavoro viene compiuto utilizzando il software DGIndex, che viene chiamato tramite la classe Process dello spazio dei nomi System.Diagnostic del framework .NET, che fornisce l'accesso ai processi locali e remoti e consente di avviare e arrestare i processi locali del sistema, utilizzando la seguente sintassi:

DGIndex.exe -IA=6 -FO=0 -OM=2 -IF=["input"] -OF=["output”] -exit

le cui opzioni hanno il seguente significato:

- **-IA=6:** selezione dell’algoritmo iDCT (inverse discrete cosine transform) utilizzato nella decodifica ed indicizzazione del video, nel nostro caso si utilizza l’algoritmo 32-bit SSEMMX (Skal).
- **-FO=0:** specifica il tipo di Field Operation (gestione dei flag RFF) nel particolare l’opzione 0 consiste nell’Honor Pulldown Flags, che effettua la ripetizione del field quando tale flag è presente; ciò significa che il video in ingresso al frameserver apparirà esattamente come inteso nella periferica di visualizzazione. Se si verifica un 3:2 pulldown nella clip si avrà la ripetizione standard di tre frame progressivi e di due interlacciati.
- **-OM=2**: seleziona il metodo di output dell’esecuzione, nel nostro caso si utilizza il metodo 2=Demux All Tracks, che effettua l’estrazione di tutte le tracce audio in file separati nella directory di output.
- **-IF:** percorso del file vob di input.
- **-OF:** indica il percorso in cui saranno salvati i file in output.
- **-exit:** termina l’esecuzione di DGIndex al termine del processo.

Una volta terminata questa fase nella directory di lavoro sono presenti il file d2v che indicizza il video ed almeno un file .ac3, che rappresenta il flusso audio del filmato, nel nostro caso nei video è presente una sola traccia audio. La seconda fase della conversione consiste nel trasformare l’audio in formato ac3 estratto dal DVD in una formato non compresso (file wav), per ottenere tale risultato si utilizza il metodo ConversioneAudio1() che richiama il software Azid, in maniera simile al metodo precedente, in questo caso nella linea di comando si specificano solo il percorso ed il nome dei file di ingresso e uscita, per il resto si sfruttano le seguenti opzioni di default:

- **-ezero:** opzione per la gestione degli errori, se in un frame si verifica la presenza di un errore esso viene saltato e non codificato.

- **-mstereo:** specifica il tipo di decodifica da utilizzare per se lo stream contiene due canali audio mono (chiamato 1+1), nel nostro caso assegna il primo al canale sinistro ed il secondo a destra.
- **-ssurround:** indica il metodo di downmix stereo quando è selezionata l'opzione 2/0.
- **-d2/0:** quest'opzione seleziona il numero di canali frontali e posteriori, in questo caso si utilizzano sono due canali frontali.
- **-ol,r:** indica quali dei sei canali del formato ac3 vengono inviati in output, dato che si usa l'opziona -d2/0 si usano di default solo il canale destro (r) e sinistro (l).
- **-l0:** controlla il downmixing del canale LFE del formato ac3, nel nostro caso esso non viene effettuato.
- **-Cbsi:**quest'opzione controlla il livello centrale del downmix nei canali LR, si utilizza il valore ricavato dal BSI (*bit stream information*, la sezione del frame in cui sono contenute le informazioni) del frame ac3.
- **-Sbsi:** quest'opzione controlla il livello del downmix di surround nei canali LR, anche in questo caso si utilizzano le informazioni contenute nei BSI.
- **-cnone:** indica se e quale tipo di compressione dinamica viene utilizzato.
- **-g1:** guadagno, in dB assegnato ai canali in uscita.

Una volta terminata questa esecuzione nella cartella di lavoro viene salvato un file wav contenete la versione PCM del flusso audio, a questo punto viene invocato il metodo ConversioneAudio2() per effettuare la seconda parte della transcodifica, che genererà il file mp3 che poi verrà incapsulato nel filmato finale; anche in questa fase viene chiamato un processo esterno, l'encoder audio Lame, di cui utilizziamo il metodo CBR (constant bitrate) a 128kbps. L'ultima parte del lavoro consiste nel metodo Esecuzione(), nel quale viene creato uno script per Virtualdub di cui in seguito mostriamo i comandi principali:

```
VirtualDub.Open("C:\\GRID\\Test\\ScriptAvisynt.avs",0,0);
VirtualDub.video.SetMode(1);
VirtualDub.video.SetCompression(0x64697678,0,10000,0);
[...]
VirtualDub.stream[0].SetMode(0);
VirtualDub.stream[0].SetSource("percorso del file audio",0x00000202,1);
VirtualDub.stream[0].SetInterleave(1,500,1,0,"delay”);
[...]
VirtualDub.SaveAVI("\\\\cl001000090003\\GRID\\Test\\Nome_File");
```

Le principali opzioni di configurazione principali si dividono nella prima parte in cui vengono settati il metodo di processing video (Fast Recompress) e viene scelto il codec video adeguato, nel nostro caso XviD. Poi c'è la parte successiva in cui

vengono definite le opzioni sullo stream audio, in questo caso il metodo di processing consiste nella copia del flusso (Direct Stream Copy), dato che è già disponibile il file audio nel formato desiderato, in seguito tale file viene importato dal software e viene multiplexato assieme al video; molto importante è il comando successivo che ha il compito di definire le opzioni di interleaving, cioè la tecnica utilizzata per disporre i dati in maniera non contigua, al fine di migliorare le prestazioni in caso di errori e effettuare una corretta sincronizzazione tra audio e video, ed in particolare il ritardo da assegnare al flusso audio rispetto a quello video, tale dato viene fornito direttamente nel nome del file ac3 al momento della sua creazione. Il comando finale consiste nel definire il percorso in cui salvare il file creato, da notarsi come viene specificato un indirizzo di rete, in modo da salvare il file direttamente nel nodo centrale così da evitare la necessita di effettuare una copia , diminuendo così l'overhead del processo. infine importante da notare come l'input del software non sia un file video ma uno script di Avisynth, utilizzato per aprire il video in modalità frameserving e per eseguire alcuni processi di editing, in particolare lo script si occupa di due aspetti fondamentali, calcolare l'aspect ratio, la larghezza e l'altezza del filmato in base all'originale in modo da effettuare correttamente il cropping ed il resizing del video e compiere il deintelacing in modo da rendere il filmato in modalità progressiva, in aggiunta utilizza una filtro per l'eliminazione del rumore. Una volta creato lo script il metodo non lancia direttamente il software VirtualDub, ciò perché tale programma necessita l'accesso al desktop per l'interfaccia grafica, cosa che non è possibile alle applicazioni avviate tramite i task grid; per risolvere questo problema è stato sviluppato un servizio windows, i windows services sono programmi che girano in background all'interno del sistema operativo ed in genere rappresentano un valido e solido sistema per effettuare operazioni asincrone che possono servire in qualsiasi tipo di applicazione e possono interfacciarsi con l'ambiente di lavoro desktop risolvendo il nostro problema. Il servizio viene attivato su tutte le macchine in cui può potenzialmente essere eseguito il task, esso non fa altro che mettersi in pausa attenendo di essere risvegliato dal processo principale tramite la classe ServiceController dello spazio dei nomi System.ServiceProcess, nel momento in cui viene attivato manda in esecuzione Virtualdub, ed una volta che esso ha terminato il lavoro ritorna in stato di pausa. Utilizzando questo sistema si ottiene un risparmio sull'uso delle risorse poiché il servizio per tutto il tempo in cui non è necessario si trova sospeso, quindi non con

il minimo consumo possibile, inoltre l'uso dei metodi della classe ServiceController per attivarlo o disattivarlo, evita la necessità di implementare un sistema di comunicazione tra servizio e processo a scambio di messaggi o a memoria condivisa, diminuendo ulteriormente l'uso di risorse.

5.1.3 Risultati ottenuti

Per testare le performance del sistema sono stati eseguiti comparando i tempi necessari per ottenere la copia del file convertito nel formato XviD tramite l'architettura distribuita rispetto a quelli ottenuti utilizzando il software AutoGK, che compie lo stesso procedimento precedentemente descritto su una macchina singola; il filmato fornito consiste in tre file vob, i primi due (VTS_01 e VTS_02) delle dimensioni di 1 GB, che contengono il grosso del filmato ed il terzo file in cui è memorizzato l'ultimo minuto della di dimensioni 18.7 MB. Oltre ai test effettuati sui singoli file, abbiamo utilizzato anche un file delle dimensioni di 500 MB ottenuto dalla divisione in due del primo, infine è stato effettuata una prova con tutti e tre i file in modo da ricostruire l'intero filmato; il software necessario è stato installato su quattro macchine diverse, una delle quali fa girare anche l'applicazione centrale, ciò permette di avere fino a quattro processing in parallelo. Di seguito sono riassunti i vari risultati ottenuti su una media di cinque prove per ogni categoria di dimensioni e l'analisi dettagliata di un particolare caso con un file di 1GB tramite i log dei due sistemi.

Dimensioni	Tempo Medio Sistema	Tempo Medio AutoGK	Tempo Massimo Sistema	Tempo Massimo AutoGK
18,7 MB	2 min, 27 sec	2 min, 04 sec	5 min, 13 min	2 min, 15 sec
500 MB	48 min, 31 sec	57 min, 46 sec	54 min, 31 sec	1 ora, 02 min
1 GB	1 ora, 56 min	2 ore, 12 min	2 ore, 57 min	2 ore, 25 min
2 GB	3 ore, 16 min	4 ore, 32 min	5 ore, 23 min	4 ore, 51 min

LOG del sistema
9:22:00 Inizio Processo
9:31:58 File Diviso
9:32:28 Processi in coda GRID
9:45:00 Inizio Esecuzione
9:51:40 Fine DGIndex
9:53:12 Fine Azid
9:55:05 Fine Lame
10:26:35 Fine Virtualdub
10:39:57 Fine ricostruzione avi

LOG di AutoGK
[24/07/07 11.03.03] Job started.
[24/07/07 11.03.03] Started encoding.

[24/07/07 11.03.03] Demuxing and indexing.
[24/07/07 11.17.31] Analyzing source.
[24/07/07 11.25.35] Decoding audio.
[24/07/07 11.27.51] Normalizing audio.
[24/07/07 11.29.12] Encoding audio.
[24/07/07 11.40.58] Running single pass encoding.
[24/07/07 13.09.42] Speed was: 17,17 fps.
[24/07/07 13.09.42] Job finished. Total time: 2 hours 6 minutes 38 seconds

Dai risultati si nota come in generale il processo distribuito ottiene mediamente tempi migliori nonostante l'overhead introdotto dall'aggiunta dei tempi necessari alla divisione del file sorgente ed alla ricostruzione del file prodotto; tuttavia tale sistema soffre una maggiore variabilità nei risultati che nel caso peggiore risultano addirittura peggiori rispetto ad AutoGK, tale situazione è dovuta all'impredicibilità intrinseca nella gestione dei task grid che non forniscono alcuna garanzia sul tempo di completamento, ad esempio una delle macchine potrebbero essere inattive o eccessivamente cariche il che influisce sulla tempistica. Più nello specifico nei primi due casi di studio menzionati l'architettura ha generato due task di esecuzione dato che le dimensioni del file di partenza erano minori di 50MB, nel terzo caso sono stati generati quattro task che, in condizioni ideali, vengono tutti eseguiti contemporaneamente riducendo al minimo i tempi di esecuzione; infine l'ultimo caso porta alla creazione di ben 9 task distinti, ma di essi possono esserne eseguiti solo quattro alla volta.

5.2 Architettura per lo Streaming

5.2.1 Approccio al problema

Il secondo problema affrontato riguarda la ricerca di una migliore gestione dei flussi di streaming video nella rete interna del Consorzio per evitare la saturazione della banda disponibile. Tale problema nasce dal fatto che la rete del consorzio ha una struttura a stella, in cui è presente un nodo centrale che contiene il server di streaming che riceve le richieste dai nodi esterni ed invia i flussi corrispondenti; nel caso in cui più nodi appartenenti alla stessa sotto rete debbano visualizzare gli stessi contenuti nel modello attuale inviano separatamente una diversa richiesta al server centrale che invia lo stesso stream separatamente per ciascun nodo effettivamente duplicando il flussi di dati e di conseguenza la banda necessaria che cresce linearmente con il numero di nodi, che se abbastanza elevato può portare facilmente alla saturazione del canale impedendo il corretto funzionamento dei sistemi di rete ed una bassa qualità negli streaming in arrivo.
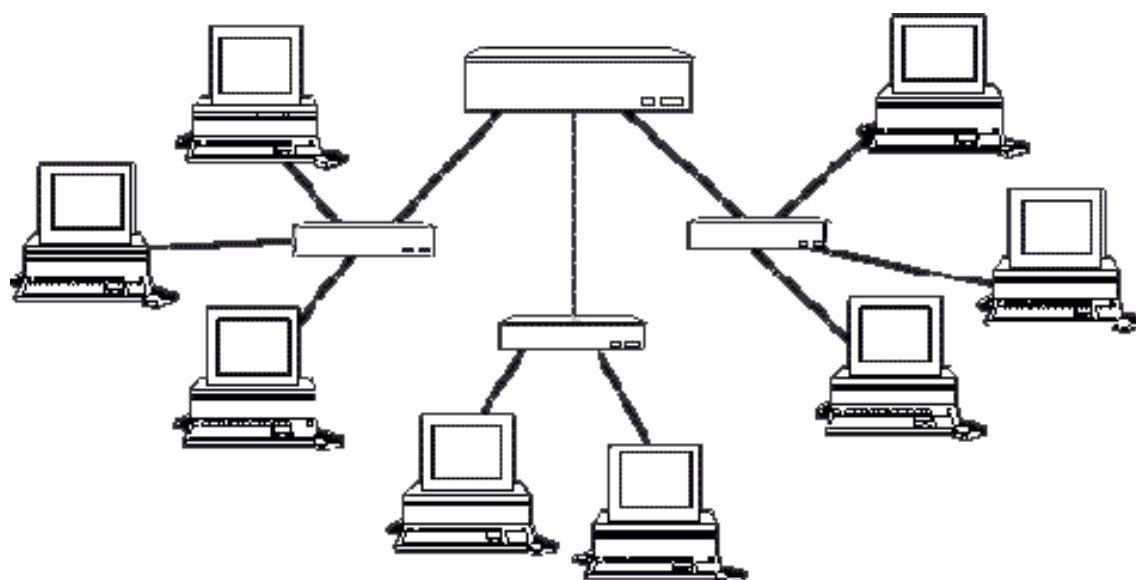
Immagine 5.5- Esempio di rete a stella

Il metodo utilizzato per questo sistema consiste nello sviluppo di un architettura che si occupi della gestione delle domande e delle richieste degli streaming in modo da evitarne, quando possibile, la duplicazione; ciò è risolto in due fasi, per prima cosa si crea un punto di accesso intermedio a cui vengono effettivamente inoltrate le richieste di streaming, che poi effettuerà la richiesta all'indirizzo corretto, a questo punto lo streaming viene inviato al nodo intermedio che si occuperà poi della ritrasmissione in broadcast a tutti i nodi che ne fanno richiesta; la seconda è un'applicazione ad hoc per l'invio e la ricezione degli streaming, piuttosto che utilizzare direttamente il software di riproduzione video (tipicamente Windows Media Player), in modo da rendere trasparente la modifica degli indirizzi di streaming.

5.2.2 Sistema Sviluppato

L'architettura progettata per è formata da due parti, una parte server, il cui compito è gestire le richieste che gli arrivano, verificare se i contenuti sono disponibili in locale, ed eventualmente inoltrarle, e soprattutto ha il compito di evitare la duplicazione non necessaria degli stream. L'altra parte del progetto consiste in un client, che si occupa di inoltrare le richieste al server e di visualizzarle tramite il media player, tale programma ha anche diverse opzioni per la gestione del flusso in entrata che permettono eventualmente di salvare il video in un file avi o di ritrasmetterlo tramite un indirizzo pubblico così da permettere la creazione di una rete di streaming Peer to Peer.

5.2.2.1 Client P2P

Di seguito mostriamo l'interfaccia del programma client e ne descriviamo le caratteristiche e le opzioni principali, tale interfaccia è divisa in due tabelle; la prima permette di richiedere al server gli stream disponibili e selezionare quelli desiderati da scaricare , nella seconda sono disponibili diverse opzioni per la ricezione dei flussi ed un opzione per collegarsi ad uno stream locale piuttosto che al server.

- ***Richiedi URL specifico:*** se questo checkbox è selezionato il text box accanto viene attivato, in tale modalità la richiesta di uno stream avviene tramite l'indirizzo specificato, sovrascrivendo eventuali selezioni delle liste di stream disponibili nel server.
- ***File Disponibili:*** in questa listbox vengono elencati tutti i file disponibili nel server a cui il client è collegato. Tale sezione viene aggiornata sia all'attivazione del processo che alla selezione del comando aggiorna.
- ***Stream Attivi:*** simile alla lista precedente contiene l'elenco degli stream attualmente inviati dal server a cui è possibile collegarsi direttamente; tra le due liste è possibile selezionare solo un singolo oggetto.
- ***Aggiorna:*** quando questo button è premuto viene inviata una richiesta al server che risponde inviando la lista aggiornata dei file disponibile e degli stream attivi.
- ***Richiesta Stream:*** questo comando invoca il metodo invia al server la richiesta dello stream desiderato, la richiesta può essere uno degli elementi selezionati nelle liste o uno stream particolare di cui deve venire specificato l'URL se l'opzione è attivata.
- ***Visualizza lo Stream:*** se questa opzione è selezionata il flusso viene semplicemente visualizzato, senza altre manipolazioni.
- ***Visualizza e Salva:*** questa opzione permette di visualizzare lo stream e contemporaneamente di salvarne una copia locale nel percorso specificato nel textbox accanto.
- ***Visualizza e Ritrasmetti:*** l'ultima opzione fa si che il client agisca da ripetitore e ritrasmetta lo stream; il flusso viene pubblicato con l'indirizzo di rete locale del computer su cui il programma è eseguito e con la porta specificata nel textbox.
- ***Richiesta Stream Locale:*** questa opzione permette al client di collegarsi ad uno stream ritrasmesso da un altro client una volta specificato l'indirizzo e la porta adeguati.
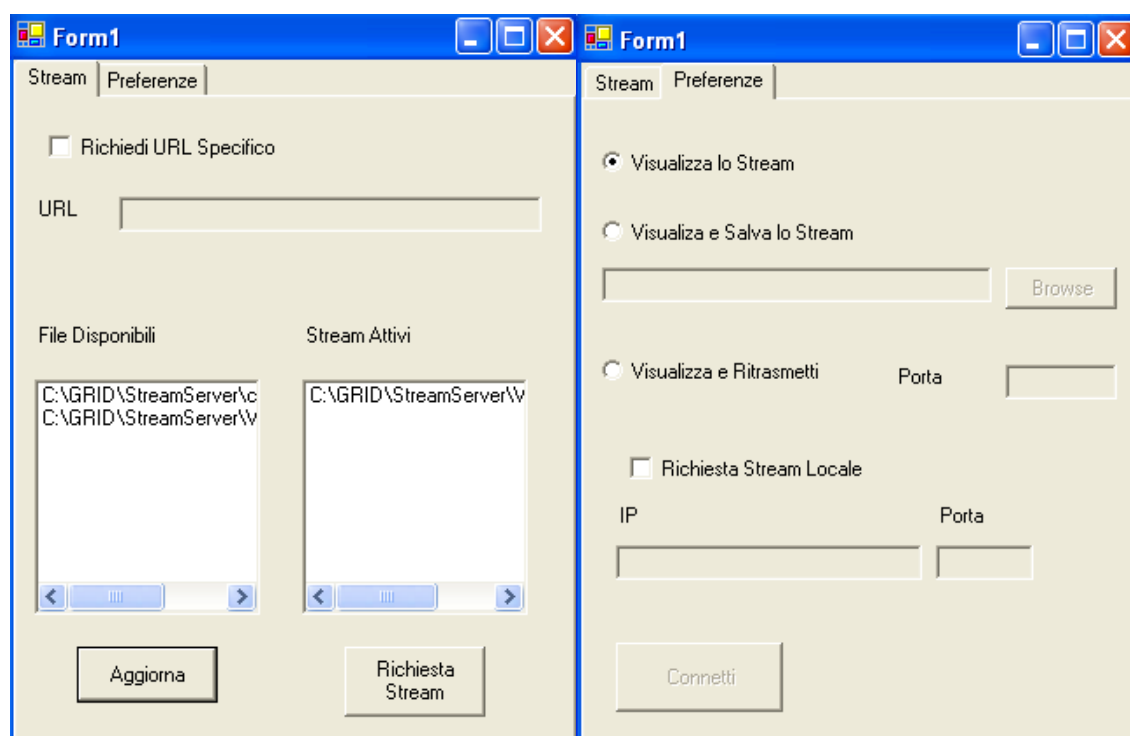
Immagine 5.6- Interfaccia del ClientP2P

Le operazioni principali del programma consistono nell'invocazione di due metodi, lo StartClient(string StreamURL) che crea il client TCP che si occupa di comunicare con il server in caso di una richiesta di aggiornamento della lista dei contenuti disponibili o dell'effettiva richiesta di uno streaming, e di ricevere e processare le informazioni relative, nel primo caso distinguendo tra i vari messaggi per elencare correttamente le informazioni ricevute tra i file disponibili o tra gli stream attivi, nel secondo caso riceve l'informazione necessaria alla ricezione dello streaming video, la memorizza, ed a quel punto invoca il metodo StreamIn(). Tale metodo si occupa di avviare il software VLC per ricevere e visualizzare il flusso video, il processo utilizza i seguenti comandi in base all'opzione effettivamente selezionata.

```
"vlc http://"+IP+":"+Port+" vlc:quit"
```

Questo comando semplicemente apre VLC e visualizza lo stream in base alle informazioni ricevute dal server, senza alcuna modifica, ed una volta terminato chiude il programma.

```
"vlc http://"+IP+":"+Port+" :sout=#duplicate{dst=display,dst=std
    {access=file,mux=ts,dst="+textBox4.Text+"}} vlc:quit"
```

Questo comando corrisponde all'opzione di salvataggio del filmato, VLC invia in output due flussi distinti ottenuti dalla duplicazione del flusso in arrivo; il primo viene riprodotto, il secondo viene salvato in un file di cui viene specificata il percorso tra le opzioni del programma. Il file viene salvato nello stesso formato di ingresso,

nel nostro caso l'MPEG-2, tra le opzioni è specificato il metodo di incapsulazione dei dati, cioè il tipo del contenitore multimediale (il formato del file in cui sono contenute e aggregate le informazioni audio e video per essere salvate su disco o trasmesse; il ruolo principale del container è quello di sincronizzare codec video, codec audio, sottotitoli e menu); nel nostro caso è l'MPEG-TS (*Transport Stream*) tipicamente utilizzato per conservare o inviare dati nelle occasioni in cui è probabile una perdita di dati, come l'invio attraverso un network. VLC può fornire altri tipi di formati video in cui salvare il flusso ed all'occorrenza può compiere una ricodifica dei dati cambiando il codec video o audio.

(Transport Stream)

```
"vlc http://"+IP+":"+Port+" :sout=#duplicate{dst=display,dst=std
    {access=http,mux=ts,dst="+ipAddress.ToString()+":"+textBox5.Text+"}} vlc:quit"
```

L'ultimo comando corrisponde alla funzione di ritrasmissione dello stream in ingresso, anche in questo caso il flusso audio è duplicato e, mentre uno dei due è riprodotto, l'altro viene pubblicato; tra le varie opzioni da definire per la trasmissione dello streaming ci sono la scelta del metodo di incapsulazione ed eventualmente del formato video o audio in cui convertire il filmato esattamente come per l'opzione di creazione di un file. Quello che cambia è la scelta del metodo di accesso in cui bisogna definire il protocollo di streaming output tra i seguenti:

- **HTTP:** usato principalmente nella nostra applicazione, ci si serve del metodo di streaming del protocollo HTTP (*Hyper-Text Tranfer Protocol*) usato come principale sistema per la trasmissione di informazioni sul web, nato principalmente per lo scambio dei html e file di testo al giorno d'oggi è sfruttato per trasmissioni di ogni genere di contenuto grazie alla sua flessibilità, funziona su un meccanismo richiesta/risposta (client/server): il client esegue una richiesta ed il server restituisce la risposta. Nell'uso comune il client corrisponde al browser ed il server al sito web.. Dal punto di vista dello streaming VLC può servirsi di questo protocollo per leggere uno stream da un server o, nel nostro caso, agire a sua volta da server per l'invio del contenuto multimediale sulla rete.
- **MMSH:** questo metodo emette lo streaming verso Windows Media Player utilizzando il protocollo proprietario della microsoft MMS (*Microsoft Media Services*); questo metodo funziona solo con il metodo di incapsulazione ASF (*Advanced Streaming Format*) specifico per i formati audio e video della Microsoft (Windows Media Audio e Windows Media Video). Nello specifico VLC usa una versione modificata del protocollo chiamato MMSH (*MMS over HTTP*).

- **UDP:** invia lo stream direttamente tramite il protocollo di trasporto UDP (*User Datagram Protocol*) all'indirizzo specificato, che può essere sia unicast che multicast. L'UDP ha come caratteristica di essere un protocollo di rete inaffidabile, privo di connessione, ma in compenso molto rapido ed efficiente per le applicazioni "leggere" o time-sensitive. Infatti, è usato spesso per la trasmissione di informazioni audio, video o applicazioni in tempo reale dato che spesso richiedono un ritmo minimo di spedizione e sono tolleranti a perdite di dati.
- **RTP:** utilizza l'RTP (*Real-Time Transfer Protocol*) per l'invio dei flussi e come per l'UDP può supportare sia un indirizzo unicast che uno multicast. L'RTP definisce un formato di pacchetti standardizzato per la consegna di contenuti audio e video su Internet ed è basato sul protocollo UDP, viene usato in congiunzione con RTCP (RTP Control Protocol) che monitora la qualità del servizio e trasporta le informazioni riguardo ai partecipanti ad una sessione, RTCP è sufficiente per sessioni "loosely controlled" in cui non c'è un reale controllo dei partecipanti e set-up della sessione e non è necessario che tutti i requisiti di controllo siano soddisfatti, per questo RTP può essere coadiuvato da un protocollo apposito per la gestione delle sessioni.

In tutti questi protocolli è comunque necessario specificare l'indirizzo IP e la porta su cui porsi in ascolto per ricevere il filmato, in particolare si usa l'indirizzo IP locale del client, mentre la porta viene specificata tra le opzioni. Questa caratteristica permette ai vari client di costituire una rete Peer to Peer per la condivisione dei contenuti multimediali senza necessariamente aver bisogno di collegarsi al server.

5.2.2.2 Stream Server

Lo stream server è un servizio windows che consiste in un server di rete a cui fanno riferimento un gruppo di client, dal punto di vista implementativo all'avvio del servizio vengono eseguite due operazioni, per prima si crea un arraylist interno in cui sono memorizzati i contenuti disponibili al servizio per l'invio di stream in locale; la seconda operazione manda in esecuzione un thread che esegui il metodo server(), un server TCP che attende eventuali connessioni da parte di un client; una volta effettuata una connessione il server può ricevere due tipi di richieste, una richiesta di specifica dei contenuti disponibili o una richiesta di invio dello stream. Nel primo caso il server invia al client le informazioni memorizzate riguardanti i file disponibili e gli stream attivi, nel secondo riceve l'URL del contenuto di streaming da inviare; una volta ottenuto l'url viene verificata l'eventuale presenza della ri-

chiesta tra gli streaming già attivi o meno; se lo streaming deve essere avviato da zero si invoca il metodo StreamOut(string URL) che si occupa di attivare il software VLC per inviare in uscita il flusso video, tramite il comando:

```
vlc "+URL+" :sout=#standard{access=http,mux=ts,dst=10.240.58.138:"+Porta+"} vlc:quit"
```

i dati necessari alla creazione dello stream (l'URL che identifica la locazione della sorgente video e la porta su cui il client deve porsi in ascolto per raccogliere la trasmissione), sono passati al metodo e nello stesso tempo sono memorizzati nella lista degli stream attivi; nel caso il contenuto richiesto appartiene alla lista degli streaming già attivi si recuperano i dati relativi alla trasmissione necessari al client per (tipicamente la porta), e gli vengono inviati. Molto importante nell'applicazione è la gestione della lista degli stream attivi, che avviene utilizzando una classe condivisa a cui si accede in modo protetto per tenere traccia dell'indice della risorsa all'interno della lista, così da evitare la generazione di incongruenze al termine di uno stream.

5.2.3 Risultati ottenuti

In questo caso i test sono stati effettuati solo parzialmente entro la LAN interna del laboratorio, senza aver la possibilità di ricreare esattamente le condizioni reali della rete a stella, inoltre il software utilizzato per la trasmissione e la ricezione dei flussi video (VLC) non è perfettamente compatibile con gli strumenti utilizzati nel laboratorio, in particolare il Windows Media Service ed il protocollo MMS; tali controindicazioni hanno impedito una verifica completa delle performance del programma. Per verificare il l'efficacia del sistema è stato sfruttato il servizio di prestazioni fornito dagli Strumenti di Amministrazione di Windows, che permette di monitorare le prestazioni delle varie interfaccie del sistema, nel nostro caso si sono monitorati i Byte inviati e ricevuti dell'interfaccia di rete ed i pacchetti inviati e ricevuti nel sistema tramite il protocollo di rete IP. I risultati ottenuti sono specificati nelle immagini sottostanti; la prima mostra il traffico nel caso di due client connessi allo stesso server tramite il sistema sviluppato, in questo caso il traffico si assesta intorno ai 570 KB al secondo (scala 0,0001) ed ai 410 pacchetti inviati.
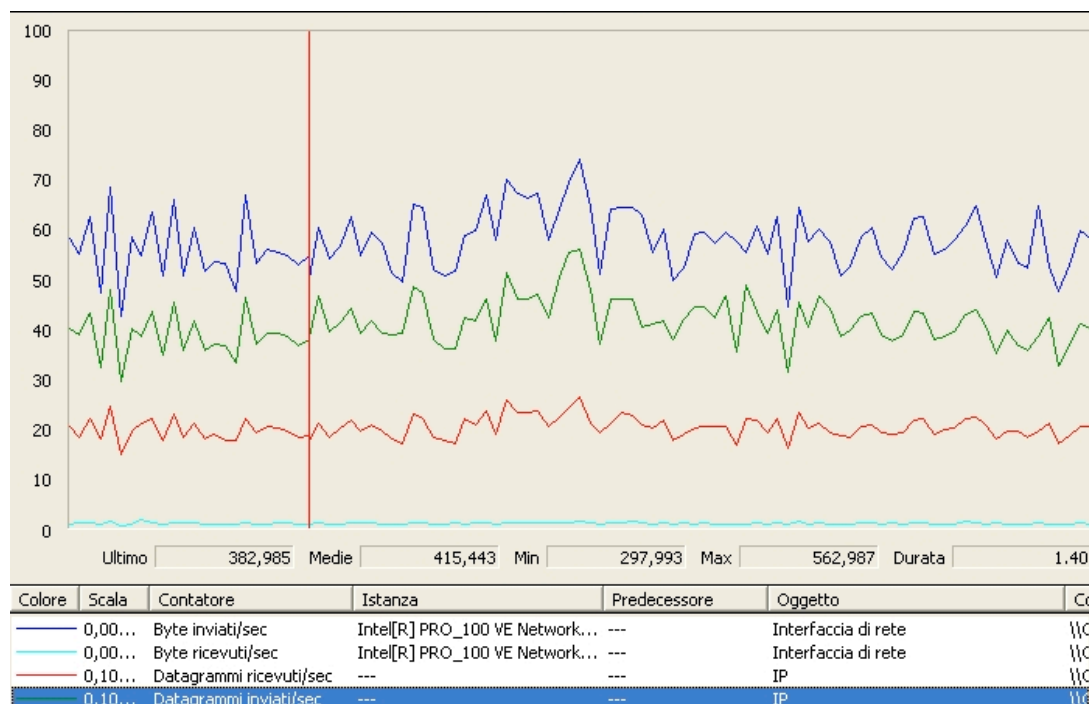
Immagine 5.7- Prestazioni Sistema

La seconda mostra gli stessi due stream inviati da due istanze separate di VLC sempre utilizzando il protocollo HTML; in questo caso il traffico di rete è estremamente più variabile e la media degli invii si assesta intorno alle 700 KB e ai 470 pacchetti, in totale il risparmio ottenuto nel caso del traffico di rete è intorno al 20%.
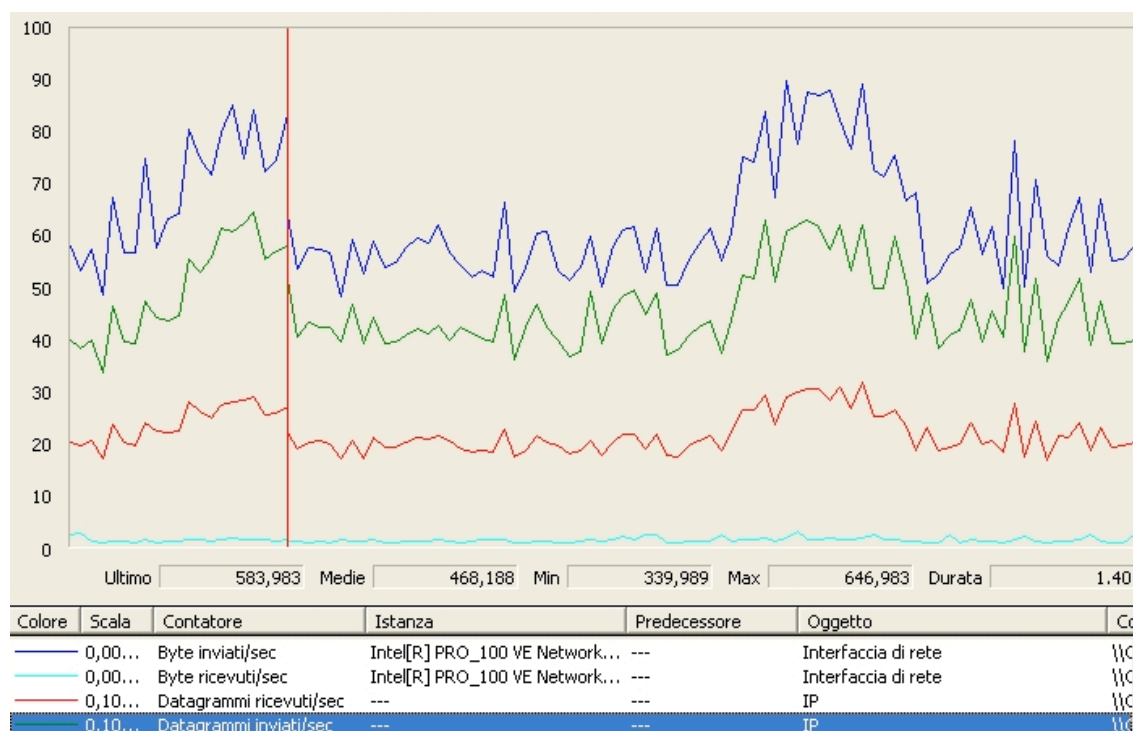
Immagine 5.8- Prestazioni Stream Separati

Conclusioni

Il primo sistema sviluppato di cui è oggetto la presente tesi si occupa della creazione di una copia di un DVD-Video effettuando una conversione dal formato originale (video MPEG-2 e Audio AC3) ad un formato diverso e più efficiente (video MPEG-4 con audio MP3), operazione che tipicamente necessita di molto tempo e di molte risorse. Tale sistema effettua la suddivisione dei contenuti multimediali da elaborare in modo da parallelizzare il lavoro in sottoprocessi di complessità minore, e sfrutta l'architettura Grid presente nel laboratorio per effettuare in modo semplice e rapido una distribuzione di tali processi in modo da gestire in modo completamente trasparente l'assegnazione dei task da eseguire ai calcolatori che al momento hanno maggiori risorse disponibili. I dati che hanno fornito la base su cui lavorare consistono su un filmato campione della lunghezza di 2:02:47 ore, e della dimensione di 2,02 GB suddiviso in tre file vob non criptati; le verifiche per il calcolo delle prestazione del sistema sono state effettuate comparando la procedura classica di conversione video sfruttando il software AutoGK ed il nostro sistema, il procedimento utilizzato da questi sistemi è molto simile e consiste nell'invocazione di una serie di strumenti esterni che forniscono le funzionalità di editing audio e video necessarie. L'architettura distribuita su cui è stato sviluppato il lavoro e condotti i test consiste in quattro nodi facenti parti dell'architettura grid su cui sono stati installati tutti gli strumenti necessari, uno di questi nodi contiene anche la parte centrale del software responsabile della suddivisione, dell'invio, del recupero e della ricostruzione delle varie parti del filmato. L'analisi dei risultati ottenuti mostra come il sistema distribuito riesce ad ottenere dei risultati mediamente migliori rispetto all'elaborazione centralizzata, ottenendo un risparmio di tempo più elevato tanto maggiori sono le dimensioni dei file da elaborare e tanti più sono i nodi disponibili; tuttavia si può notare come i tempio del sistema abbiano una varianza estremamente maggiore rispetto al singolo processo, dato che la gestione dell'esecuzione dei task distribuiti è totalmente demandata all'architettura grid che, se da un lato permette di specificare con estrema completezza i parametri per l'algoritmo di assegnazione dei task alle macchine, dall'altra non offre nessuna garanzia sull'effettivo tempo di esecuzione degli stessi, ch epuò essere estremamente variabile a seconda delle condizione interne del nodo. Nonostante ciò appare co-

munque preferibile l’uso di tale sistema dato che, anche nei casi peggiori, permette una distribuzione del carico di lavoro su macchine altrimenti inutilizzate o sotto-utilizzate. Dal punto di vista di sviluppi successivi la base fornita dal nostro sistema è ampiamente soggetta ad una serie di modifiche ed estensioni, in particolare nel caso delle eventuali configurazioni dei software di editing video, dato che nella versione attuale il processo viene eseguito con una configurazione standard che potrebbe essere decisamente migliorata; alcuni suggerimenti su tale sviluppo comprendono l’inclusione di editor per lo scripting di Avisynth e di VirtualDub, che sono i software che effettivamente compiono la manipolazione del filmato, o l'inclusione di un’interfaccia grafica che permette la selezione di varie opzioni che si tradurranno in maniera trasparente sulla creazione delle linee di commando nell’invocazione dei vari programmi.

Il secondo sistema tratta il problema della gestione degli streaming video nella rete interna del Consorzio, problema generato dalla duplicazione dei vari flussi in uscita causata da più richieste contemporanee dello stesso contenuto da parte di computer appartenente alla stessa sottorete. In questo caso i test sono stati effettuati servendosi degli stessi filmati utilizzati per la conversione video, che tramite il software VLC sono stati trasmessi come flussi di streaming tra i vari computer della rete locale del laboratorio; in tal modo abbiamo confrontato le statistica sull’occupazione della rete nel caso dell’uso dell’architettura di gestione e nel caso di trasmissioni dirette senza alcun supporto. In questo caso i risultati ottenuti sono stati positivi, ed in generale si è riusciti ad ottenere un traffico di rete inferiore di circa il 20% con l’uso del sistema rispetto al non farne uso. Un altro vantaggio dell’architettura sviluppata consiste nel supporto presente alla richiesta diretta di streaming senza necessità di passare attraverso il server, cosa che permette una maggiore flessibilità del nostro programma; tale supporto potrebbe essere ampliato in futuro implementando una comunicazione tra i veri server ed una policy per la gestione delle richieste in modo da inviarle ai server, o ai client che fungono da ripetiori, più adatti, rendendo l’architettura simile ad una rete Peer to Peer.

Bibliografia

[1] Iain E.G. Richardson ***H.264 and MPEG-4 Video Compression*** *video coding for next generation Multimedia*

[2] John F. Koegel Buford ***Multimedia Systems***

[3] Philip Shaddock ***Multimedia in Pratica***

[4] John A. Waterworth ***Multimedia Tecnologia e Applicazioni***

[5] Ralf Steinmetz & Klara Nahrstedt ***Multimedia Computing, Communications & Applications***

[6] Atul Puri & Tsuhan Chen ***Multimedia Systems, Standards & Networks***

[7] Jerry D. Gibson, Toby Berger, Tom Laskabaugh, Dave Lindberg & Richard L. Baker ***Digital Compression for Multimedia*** *principles and standards*

[8] Michael Tapic ***Streaming Media Demistified***

[9] John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee ***Video Streaming: Concepts, Algorithms, and Systems***

[10] Advanced Television System Committè ***Digital Compression Standard (AC3)***

[11] Syed Ali Khayam ***The Discrete Cosine Transform (DCT): Theory and Application***

[12] Mauro Migliardi ***GRID Computing***

[13] P.N. Tudor ***MPEG-2 VIDEO COMPRESSION***
(http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml)

[14] Roberto Cappuccio, Andrea Testarmata, Raffaele Sgherri ***Avanade Grid Architecture 2.5 - Blueprint***

[15] Andrea Testarmata, Raffaele Sgherri ***Avanade Grid Architecture 2.5 - Design Documentation***

[16] Andrea Testarmata ***Avanade Grid Architecture 2.5 - Developer's Documentation***

[17] Wikipedia (www.wikipedia.org)

[18] Doom9 (www.doom9.org)

[19] DivXITA (www.divxita.it)

[20] AviSynth Manuale Utente

[21] DG Manuale Utente

[22] Azid Manuale Utente

[23] Lame Manuale Utente

[24] VLC Manuale Utente